Explainable Trajectory Prediction
(In collaboration with Euranova)

Atefeh Bahrami

August 2023

Abstract

This thesis delves into the burgeoning field of trajectory prediction, placing particular emphasis on the vital role of explainability. At the heart of our investigation lie two predictive tasks: pinpointing a moving object's ultimate destination and forecasting its next stop mere minutes before arrival. Recognizing trajectories as sequences, we harnessed the power of Long Short-Term Memory (LSTM) networks. By experimenting with varied learning techniques, we adeptly addressed both tasks, achieving commendable accuracy. Notably, when tasked with predicting a destination 2.5 minutes in advance, our model showcased a commendable accuracy of **47.92%**. This suggests that as a vehicle is close to its destination, our model can predict its final stop with nearly **48%** precision 2.5 minutes before.

To further the study's depth, we sought to explain our models. We leveraged two explanatory techniques: the LIME text explainer and the disturbance approach. Our explorations revealed that the LSTM model, when predicting the final destination, predominantly relies on the trajectory's recent points rather than its entirety. Armed with this understanding, we refined our model's learning approach, enhancing its predictive prowess for earlier destination estimations. Notably, both our explanation methods consistently highlighted the significant influence of the most recent trajectory points on next-position predictions.

Keywords— Explainable Trajectory Prediction, Sequential data, Explainability, interpretability, LSTM, Map-Matching

Contents

List of Figures

List of Tables

Acknowledgements

Life's journey is never a straight path but a series of zigzagging roads, each diverging into myriad directions. Sometimes, the less-trodden path calls out to us, whispering of unexplored landscapes and new perspectives. For me, that path began in high school, when the world of Computer Science unfurled before my eyes. That initial spark has guided me through a labyrinth of challenges, discoveries, failures, and triumphs, shaping my reality and opening my mind to endless possibilities.

Here I stand, at the culmination of a two-year academic expedition, yet the echoes of the journey span far more than just this period. The path has been arduous, with moments of both triumph and despair. However, these experiences have imprinted invaluable lessons, instilled resilience, and nurtured a deep-rooted love for my field. As I pen these acknowledgments, my heart brims with gratitude for those who accompanied me, brightening even the darkest corners of this expedition.

First and foremost, I want to express my most profound appreciation to my thesis advisors, Dr. Siegfried Nijssen and Mr.Sabri Skhiri. Their steadfast support, intellectual insight, and unyielding dedication have guided me through the tangled web of academia, pushing me to constantly broaden my horizons and never shy away from a challenge.

To my beloved husband, Nima Farnoodian, your unwavering belief in me and ceaseless love have been my most significant source of strength. Your understanding, patience, and encouragement have been my guiding stars, inspiring me to keep going, even when the path seemed steep and daunting.

To my parents, Fatemeh and Ali, and my brothers, Hossein, Hassan, and Hamed, I owe you a debt of gratitude that words can hardly convey. Your unwavering love and relentless faith in my abilities have been instrumental in shaping me into the woman I am today. Your sacrifices, teachings, and endless support have been the bedrock of my journey.

My wonderful friends, Andia Danlee Fathi, Dr. Yann Danle, Mojgan Giahi, and my other friends and classmates as my extended family, have each added color and warmth to this journey. Your constant companionship and unending moral support have been my refuge during challenging times. Each one of you has, in your unique way, made this journey even more memorable.

Lastly, to the committed readers and peers who took the time to delve into my work, your engagement adds immeasurable value to the academic world. This thesis doesn't merely sit in solitude but thrives from your insights, critiques, and acknowledgment. I'm deeply thankful for your time and effort, which greatly contribute to the ever-evolving realm of research.

This thesis is a testament to my work and reflects all the love, support, and encouragement I've received from each of you. I am forever indebted to you all for your contributions that have made this journey not just possible but genuinely enriching and unforgettable. Thank you.

Chapter 1

Introduction

In the vibrant city of Porto, taxis weave through the cobblestone streets, an integral part of the city's bustling transportation network. It's 5:30 PM on a Friday, and the historic center of Porto is alive with locals and tourists alike, all eager to get to their destinations. Among them is Alex, a seasoned taxi driver familiar with every nook and cranny of the city. He's just received a hail from a passenger, Jamie, who needs to catch a flight but hasn't specified the airport. Alex's onboard navigation system, equipped with the latest in trajectory prediction, instantly suggests: "Based on the current route and patterns of other taxis, it's likely your passenger is heading towards Francisco Sá Carneiro Airport. Would you like the quickest route?"

Jamie, taken aback, confirms that Francisco Sá Carneiro Airport is indeed where he's headed. This trajectory prediction isn't derived from Jamie's past rides or habits; it's constructed from analyzing patterns of countless taxis navigating Porto, discerning which routes most taxis take to popular destinations like the airport. Jamie asks, "How did the system figure that out?"

In today's modern world, the continuous evolution of mobility systems in vehicles has seen the integration of sophisticated algorithms and cutting-edge technologies, all tailored to make commuting smoother and more intuitive. Amid the myriad of solutions and innovations introduced, one intriguing facet that could unravel the mystery of Jamie's question is the domain of Trajectory Prediction. This realm delves deep into understanding the paths taken, not by probing into individual habits, but by observing the collective patterns of many. It's akin to capturing the essence of a thousand journeys to predict just one.

In this thesis, undertaken in collaboration with Euronova company, one important aspect is focusing on trajectory prediction, emphasizing two critical tasks: first, predicting the ultimate destination of a moving object, and second, estimating its destination shortly before it reaches its final point. To tackle these challenges, we employed multiple strategies. Given that trajectory data inherently embodies sequential information, our foremost method centers on using Long Short-Term Memory (LSTM) networks. By tweaking the learning techniques within LSTM, we achieved commendable accuracy in predicting both tasks.

In our study, the emphasis doesn't merely rest on achieving accurate trajectory predictions. Equally vital is the quest to understand and explain our models' underlying decisions. Since machine learning models grow in complexity, explainability emerges as an equally crucial yet often overlooked element. Particularly when delving into areas that hold

real-world implications, such as trajectory prediction, understanding a model's inner workings and reasoning isn't just academic; it's imperative. Being able to interpret and explain a model's decisions provides transparency, fostering trust, ensuring ethical implications are addressed, and ultimately enhancing the model's usability and applicability.

The remainder of the thesis is structured as follows. Chapter 2 provides the background for foundational concepts of responsible AI, explainable AI (XAI), and trajectory data analysis. As this thesis focuses primarily on trajectory prediction, chapter 3 examines some predicting models including LSTM and n-gram models for trajectory prediction. In Chapter 4, two approaches for explaining the models are presented, and compared the result. In the final Chapter 5, the conclusion and future work are presented.

Chapter 2

Background

Chapter 2 delves into the foundational concepts underpinning responsible AI, explainable AI (XAI), and trajectory data analysis. Responsible AI practices are explored, emphasizing their role in fostering ethical, transparent, and accountable AI systems that align with stakeholder expectations and regulations. The significance of XAI in enhancing human understanding of complex AI models is examined, introducing methods like LIME and SHAP. The fundamental distinctions between interpretability and explainability are outlined. The chapter further categorizes XAI methods based on the types of explanations they offer, setting the stage for in-depth exploration of renowned methods. Finally, the collection sources, pre-processing techniques, and trajectory modeling methods for trajectory data are discussed, highlighting their importance in unraveling valuable insights for urban planning and transportation engineering.

Artificial Intelligence (AI) 's impact on human endeavors and business operations is undeniable in today's world. While extensive contemporary literature examines the numerous advantages that can arise from AI technologies, there is a pressing need to explore ways AI can function responsibly. It is crucial to ensure that AI operations align with the expectations of stakeholders and comply with relevant regulations. The gap is sought to be filled by investigating the practices of responsible AI and identifying the potential advantages that may arise from the implementation of such responsible AI practices.

Responsible AI operates as a governing structure to orchestrate, apply, assess, and supervise AI machinery to unlock new prospects for improved service delivery. It emphasizes the creation and execution of ethical, transparent, and accountable AI systems that aid in preserving trust at an individual level and reducing the invasion of privacy. Responsible AI situates humans (for instance, end-users) at its core and satisfies expectations from stakeholders, adhering to applicable rules and regulations. Prior to architecting and actualizing responsible AI, organizations must comprehend the practices that will assist in fostering the ethics and trust linked with AI usage. The quartet of responsible AI practices consists of (1) Governance of data; (2) Design of ethical solutions; (3) Human-centric surveillance/risk management; and (4) Training and Education. The real-world instances of responsible AI exhibit these practices[1]. Data governance of responsible AI mainly focuses on transparency, trust, and explainability which is the goal of this study.

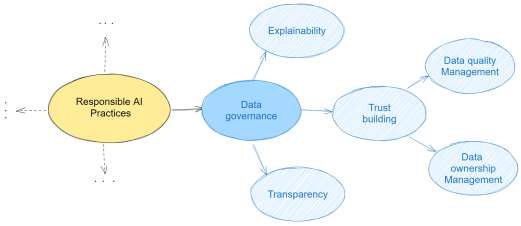

Figure 2.1: An adapted concept map of responsible AI practices focusing on data governance and explainability.[1]

2.1 Explainable AI (XAI)

In recent years, models based on artificial intelligence (AI) have emerged as pivotal technology, facilitating groundbreaking advancements across a range of applications, including natural language processing [5], computer vision [6, 7, 8], self-driving vehicles [9], agriculture [10], and healthcare [11]. Nonetheless, the realization of these innovations has often resulted in a trade-off with model interpretability, creating difficulties for humans in comprehending the decision-making process. In this context, offering detailed and personalized explanations about the outcomes generated by AI models becomes even more significant. It has the potential to alleviate such uncertainty and enhance trust among users[12].

According to the Responsible AI framework proposed by PWC [13], interpretability and explainability are defined as the capacity to explain both the overall decision-making process and the individual predictions generated by an AI model.

This raises an inevitable question: why does explainability hold such critical importance in AI applications?

Understanding this, it becomes clear that several factors can drive the requirement for explicating AI models and their predictions. These include commercial advantages, ethical implications, or even regulatory stipulations. For instance, European Union regulation 679 [14] affords data owners the right to an explanation of decisions made using their data and allows them to challenge the decision if it results from AI models.

In sectors where decisions hold substantial impact and demand a thorough understanding, explainability not only becomes essential but also boosts the acceptance of these AI-empowered applications. This broad acceptance is necessary in situations where AI models play a supporting role, such as in medical diagnoses and circumstances where they essentially govern the decision-making process, like in autonomous driving.

Fig. 2.2 categorizes widely-employed AI models based on their complexity, potential performance in AI applications, and degree of explainability. As depicted in the figure, traditional machine learning algorithms generally offer a higher level of explainability, albeit potentially lacking in predictive performance. In contrast, more advanced algorithms, such as deep learning models, yield superior performance in complex systems but present significant challenges in terms of explainability.

2.2 Fundamental Concepts and Background

The concept of Explainable Artificial Intelligence (XAI) was initially introduced by Van Lent and colleagues in 2004. They used this term to articulate the capabilities of their system

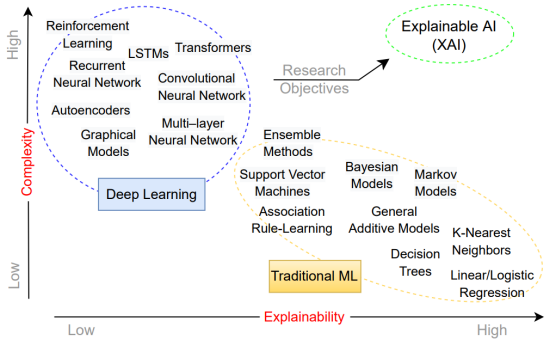

Figure 2.2: Classification of AI models according to their level of complexity, explainability, and their potential in modern AI applications[2]

that could interpret and clarify the actions of entities controlled by Artificial Intelligence within simulated gaming environments.[15]

2.2.1 Black- and White-Box Models

As shown in the fig. 2.2, some traditional machine learning models are more explainable and others are more complex and harder to interpret. Thus, one key way we categorize machine learning models is by their inherent interpretability, distinguishing between white-box and black-box models.

White-box models generally provide an understandable depiction of the relationship between input and output. For instance, in a linear regression model, the coefficients signify how a single unit change in input alters the output linearly. Other examples of white-box models include traditional machine learning models such as decision trees, rule-based learners, and k-nearest-neighbor models.

In contrast, black-box models are typically not easily understood by humans due to their high non-linearity and large model size. These characteristics allow black-box models to discern complex relationships in data that white-box models can't capture. However, this capability often comes at the expense of interpretability. Some examples of black-box models include neural networks, ensemble methods, and support vector machines.

2.2.2 Explainability and Interpretability

Interpretability and explainability are distinct yet closely related concepts in artificial intelligence. These terms are frequently used interchangeably in research, although some researchers differentiate them. Notably, they lack a precise mathematical definition and have not been quantified with specific metrics. Nevertheless, various attempts have been made to clarify these concepts [16, 17, 18], though these intuitive definitions often lack mathematical formality and rigor [19].

Interpretability is defined by Doshi-Velez and Kim [17] as "the ability to explain or present in understandable terms to a human." It's generally associated with the transparency of a model's outputs and the degree to which cause-and-effect relationships can be discerned within the system's inputs and outputs. Explainability, conversely, pertains

to the internal logic and mechanics of a machine-learning system. A model with high explainability provides a deeper understanding of the internal procedures occurring during the model's training or decision-making processes.

On the other hand, In some of the technical literature, explainability and interpretability are often used synonymously [20, 21]. Aligning with this convention, this study will also utilize both terms synonymously, recognizing their shared importance in the broader context of understanding AI models.

2.3 Categorization of the XAI methods

The landscape of interpretability methods in machine learning is broad and varied, encompassing different perspectives, such as the type of data dealt with and whether these techniques target global or local properties. The classification of these methods is multifaceted and shouldn't be viewed through a single lens, as numerous perspectives exist that further distinguish these techniques. Therefore, all aspects of each technique should be thoroughly examined to pinpoint the most appropriate method for a specific set of problem criteria. While the classification of these methods can differ across various references and papers, for the purpose of this discussion, I will be focusing on two distinct categories based on two separate papers.

The first categorization approach to interpretability methods presented by Pantelis Linardatos and colleagues [3] focuses on several distinguishing factors. This classification primarily hinges on the type of algorithms to which the methods can be applied. These are split into two categories: 'model-specific' methods confined to a particular family of algorithms and 'model-agnostic' methods universally applicable across all algorithms. Furthermore, the scale of interpretation plays a pivotal role in this categorization. This bifurcates methods into 'local' and 'global'; the former provides explanations for a specific instance, while the latter is designed to elucidate the functioning of the whole model.

Finally, the nature of the data these methods can process is also decisive. Most common methods cater to tabular data and images, although some are tailored for text data. Therefore, when utilizing this first categorization scheme, practitioners must thoroughly consider these critical factors to select the most suitable interpretability method for their needs. Fig. 2.3 provides a concise visual representation in the form of a mind map, demonstrating the various facets involved in classifying an interpretability method.

The second approach to categorization is known as XAIR [2], focusing not on providing an exhaustive list of all available explainable AI (XAI) methods but instead offering a practical guide for developers and practitioners to select suitable XAI methods. This categorization, differs from previous taxonomies that primarily categorize XAI methods on a purely functional level. XAIR introduces a categorization based on the type of explanation developers and practitioners might seek to implement to enhance their model's interpretability. It also considers the common types of machine learning models that might be used.

The factors that form the basis for this categorization are the types of explanations. Specifically, four major types of explanations are recognized:

- **Feature Importance**: Importance is given to input features such as image pixels, word tokens, or numerical features from structured data.
- **White-box models**: Methods that aim to create a transparent model that reproduces the functionality of the original black-box model.

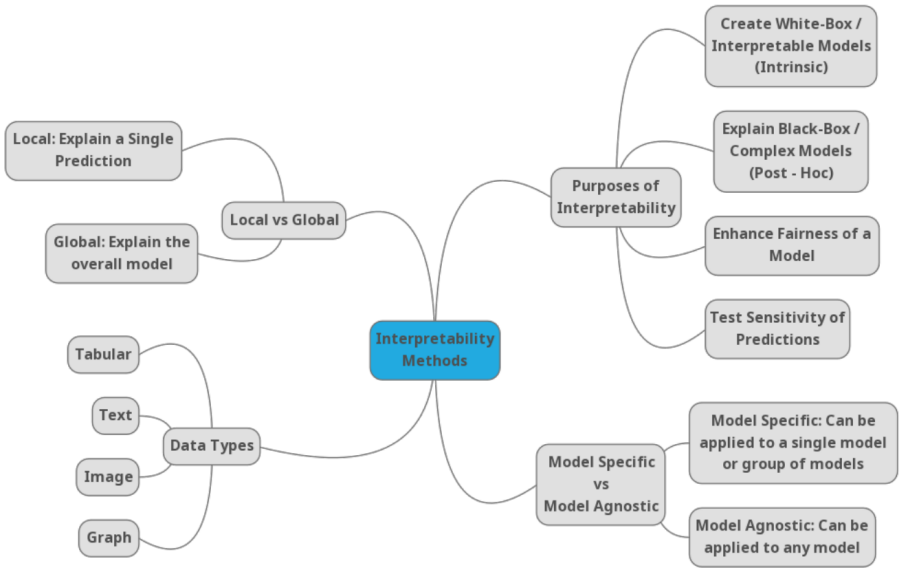

Figure 2.3: Taxonomy mind-map of Machine Learning Interpretability Techniques [3]

- **Example-based XAI**: Methods that utilize instances from the training data to clarify decisions made by the black-box model.
- **Visual explanations**: Methods that rely solely on visual explanations.

These factors create a unique approach to categorizing XAI methods, providing a simple yet effective framework for developers and practitioners to select suitable methods based on their specific needs. Table 2.1 categorizes the reported XAI methods considering mentioned factors, emphasizing their scope and functionality. Furthermore, detailed explanations of some renowned explainability methods will be provided in the forthcoming sections.

Explanation Type	Black-Box Model	Method	Scope	Functionality
Feature Importance	Any	LIME	Local	Surrogate Model
		LORE	Local	Surrogate Model
		Anchors	Local	Surrogate Model
		Occlusion	Local	Input Perturbation
		Permutation Feature Importance	Global	Input Perturbation
		Shapley Feature Importance	Global	Game-Theory
		SHAP	Both	Game-Theory
	Neural Network	Guided Backpropagation	Local	Backpropagation
		Integrated Gradients	Local	Backpropagation
		Layerwise Relevance Propagation	Local	Backpropagation
		DeepLift	Local	Backpropagation
		Testing with Concept Activation Vectors	Global	Human Concepts
		Activation Maximization	Global	Forwardpropagation
	CNN	Deconvolution	Local	Backpropagation
		Class Activation Map	Local	Backpropagation
		Grad-CAM	Local	Backpropagation
	Transformer	Attention Flow/Attention Rollout	Local	Network Graph
		Transformer Relevance Propagation	Local	Backpropagation
White-Box Model	Any	Rule Extraction	Global	Simplification
		Tree Extraction	Global	Simplification
		Model Distillation	Global	Simplification
	CNN	Attention Network	Global	Model Adaption
	RNN	Attention Network	Global	Model Adaption
Example-Based	Any	Prototypes	Global	Example (Train Data)
		Criticisms	Global	Example (Train Data)
		Counterfactuals	Global	Fictional data point
Visual Explanations	Any	Partial Dependence Plot	Global	Marginalization
		Individual Conditional Expectation	Global	Marginalization
		Accumulated Local Effects	Global	Accumulation

Table 2.1: Overview of the reported XAI methods.(Adapted from[2])

2.4 Overview of the popular XAI methods

2.4.1 LIME

One of the universal interpretability method applicable to any black-box model, known as Local Interpretable Model-agnostic Explanations, or LIME, first introduced in [22].

Garnering substantial recognition in the interpretability domain, LIME has the unique capability of decoding individual prediction scores rendered by any classifier.

The essence of LIME lies in its technique: for a specific instance and corresponding prediction, it creates simulated data by randomly sampling the surroundings of the initial input instance. This collection of 'neighbors' is then put through the original model to generate new predictions. The generated instances are assigned weights depending on their closeness to the initial input. Following this, a straightforward, comprehensible model, like a decision tree, is trained using this new dataset of tweaked instances. The local model's interpretation then serves as a window to understanding the workings of the initial black-box model.

Fig. 2.4 provides an illuminating illustration of how LIME operates to show the reasoning behind the classification of distinct components of an arbitrary image by Google's Inception neural network [23]. In this scenario, the classifier misidentifies an acoustic guitar as an electric guitar in the image. Further analysis of the LIME-provided explanation helps us understand this error: the fretboard of both the acoustic and electric guitars bear a striking resemblance. This shows how LIME can be instrumental in identifying factors that contribute to the model's decision, even if that decision may seem incorrect at the first glance.

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 2.4: Explaining an image classification prediction made by Google's Inception network, high-lighting positive pixels. The top 3 classes predicted are "Electric Guitar" (p = 0.32), "Acoustic guitar"(p = 0.24) and "Labrador" (p = 0.21) [4]

Though LIME's simplicity and power make it a standout, it's not without its shortcomings. As highlighted in 2020's first theoretical analysis of LIME [4], while the importance and value of LIME were confirmed, it was also found that inappropriate parameter selections could cause LIME to overlook significant features.

2.4.2 SHAP

Shapley Additive explanations, often referred to as SHAP [24], introduces a transformative approach to enhancing the interpretability of complex models. It's a concept born from the foundations of game theory and works by attributing importance scores to each feature involved in specific model predictions. This, in turn, significantly aids in breaking down and comprehending the black-box nature of complex models. SHAP isn't an isolated method but rather stands on the shoulders of several preceding approaches, incorporating them into a unified category called additive feature attribution. Some of the prominent methodologies in this category include LIME, DeepLIFT, and Layer-Wise Relevance Propagation. These methods, although diverse in their implementation, all converge on a common theme of interpreting model behaviour.

What sets SHAP apart is the introduction of SHAP values. These are comprehensive metrics of feature importance, possessing three distinctive attributes: local accuracy, a thoughtful approach to missing data, and consistency, which were derived in the work of [24]. Together, these properties make SHAP values a robust measure for interpreting model decisions.

However, the true ingenuity of SHAP lies in its flexible framework. The authors propose multiple methods for estimating SHAP values, demonstrating the method's versatility. Extensive experiments confirm that SHAP values excel in differentiating between diverse output classes and also align more intuitively with human rationale than many existing methodologies. This aligns with the increasing demand in AI for models that can both perform at a high level and provide understandable explanations for their decisions. On the other hand, SHAP values assume features are independent, but in real-world data, features can often be correlated especially in trajectory data which points are inherently dependent, and the value at one time step is often dependent upon with values at adjacent time steps. This can lead to misleading interpretations if not considered. There are some extended variants like KERNEL SHAP porposed in [25] which can handle dependent features.

2.5 Trajectory Data

Recent advances in sensor and mobile technology have led to an explosion in the availability and collection of urban trajectory data, specifically vehicle trajectory data. This data represents a form of time series data with spatial characteristics, consisting of sequences of time-stamped location coordinates. The surge in trajectory data demands innovative ways to efficiently manage, analyze, and extract insights from this burgeoning information source.

In this study, we focus on harnessing this complex form of data derived from vehicles, aiming to uncover rich insights that can be pivotal in urban planning, transportation engineering, and traffic prediction. It's important to note that both the temporal and spatial dependencies inherent in vehicle trajectories add to the complexity of this data, requiring sophisticated data processing and analytical techniques.

Instruments that monitor the motion of objects frequently produce an ordered sequence of points, depicting a trajectory. To define this more precisely as mentioned in [26] , a trajectory T comprises at least two points in space, and can be formally outlined as:

Definition 1. (Point) A point $p = \{x, y, t\}$ records the latitude x, the longitude y at timestamp t.

Definition 2. (Trajectory) A trajectory T is a sequence of points $\{(p_1, \gamma_1), (p_2, \gamma_2),$ $..., (p_m, \gamma_m)\}$.

In a trajectory, denoted as T, the count of data points represents its length. This length, which signifies the number of observed locations along the trajectory, provides crucial information about the detailed movements of an object. Additionally, another term, "sampling rate", is used. It defines the frequency at which data points are captured per unit of time, usually expressed in samples per second.

Besides the basic spatial and temporal information, more detailed data, represented by gamma (γ), can be associated with each data point in a trajectory. These additional information, obtained from location-based services, could include details like speed, direction, or environmental conditions at the time of data capture. Such enriched trajectory data allow us to gain a more comprehensive understanding of the moving patterns and contextual factors influencing these movements.[1]

2.5.1 Overview on Trajectory Data

Collecting trajectory data involves tracking the movement of objects over time. With the rise of modern technology, there are now a wide variety of sources from which this data can be collected. In table 2.2 some popular trajectory data collection sources, collection methods, and applications are mentioned.

2.5.2 Pre-processing of Trajectory Data

Pre-processing of trajectory data is an essential step in trajectory data analysis and management. Various methods are applied at this stage to ensure the usability and reliability

[1] For trajectories that are based solely on spatial information, we can assign gamma (γ) as an empty set, denoted as $\emptyset$. It's important to note that, alternatively, gamma (γ) could be associated with the entire trajectory as a single entity, instead of assigning it to individual data points.

Data Source	Method of Collection	Applications
Self-Driving Vehicles	Equipped with sensors that record position, orientation, and velocity.	Autonomous navigation, traffic analysis.
Traffic Cameras	Capture video footage of traffic, extract trajectories by tracking vehicles across frames.	Traffic management, urban planning.
Mobile Phones	Generate trajectory data via built-in GPS, Wi-Fi, or cellular signals.	User behavior analysis, location-based services.
Vehicle GPS	Constantly track and record geographic coordinates.	Fleet management, route optimization.
Satellites	Track movement of objects on earth, providing a global view of various phenomena.	Environmental monitoring, meteorology.
Drones	Monitor specific regions from an aerial perspective.	Surveying, disaster response.
Wearable Devices	Record trajectory data as users move.	Health monitoring, fitness tracking.
RFID and Barcodes	Track movement of goods in supply chain.	Inventory management, supply chain optimization.

Table 2.2: Trajectory Data Collection Sources, Collection Methods, and Applications

of the data. Firstly, **Trajectory Cleaning** is employed to remove noisy and redundant data, enhancing the quality of the dataset. This process includes filtering outliers or erroneous points that could skew the analysis. **Data segmentation** [27] and **calibration** [28] are two most common data cleaning techniques for trajectory data. **Synthetic Data Generation** [29] is another approach that aids in overcoming the limitations of real-world data, such as lack of variety or scarcity, by artificially creating data samples. **Data Recovery** [30] comes into play when the data collection process has gaps, often due to low sampling rates, loss of GPS signal, or sensor malfunction. This method helps in estimating and filling in these gaps, preserving the continuity and usefulness of trajectory data. **Compression** [31] is a pre-processing step focused on reducing the size of the dataset without losing significant information. It makes the storage and processing of massive trajectory data more manageable and efficient. Lastly, **Map-Matching** [32] is a critical method in cases where trajectory data needs to be aligned with a road network or geographical map. This procedure associates each point in the trajectory data to a particular location or path on the map, enabling more context-aware analyses. These methods form the foundation of effective trajectory data processing, leading to improved data-driven insights and decision-making.

Map-matching

Map-matching serves as a fundamental pre-processing measure for trajectory data by reconfiguring a raw trajectory to match an actual path within a road network accurately. This network can be depicted as a graph, either weighted or unweighted. In a weighted graph, each road segment aligns with an edge, and its weight characterizes the length

of the road. This approach aids in data cleaning by pinpointing and removing erroneous points and data compression by eliminating superfluous data points. Additionally, map-matching enhances trajectory data with the contextual information of the road network, allowing for more intricate analyses.

Many tools and libraries are available for conducting map-matching tasks, along with the **Leuven.MapMatching** library [33], there are other prominent tools such as **OpenStreetMap's OSRM** [2], **GraphHopper**[3], **Hidden Markov Map Matching Through Noise and Sparseness (HMM)** [32], and Barefoot [34]. Each offers a unique set of capabilities to cater to different aspects of map-matching, such as handling noise, accommodating data sparseness, and processing massive volumes of data.

The Leuven.MapMatching library stands out due to its robustness and precision in handling trajectory data. Its advanced functionality allows it to excel in a variety of complex scenarios, making it a preferred choice for many researchers and practitioners.

Leuven.MapMatching

The Leuven.MapMatching library employs the Hidden Markov Model (HMM) for map matching. The HMM is used to model the uncertainty between the observed GPS positions (i.e., the "emissions" in HMM terminology) and the actual location on the road network (i.e., the "hidden states"). In the HMM used in Leuven.MapMatching:

- **Hidden states**: represent the actual positions on the road network. These are the states that are estimated.
- **Observations**: represent the recorded GPS positions. These are the data we have, but they contain noise and thus do not directly give the hidden states.
- **Transition probabilities**: represent the likelihood of moving from one position on the road to another. In the context of map matching, these probabilities can be based on factors such as the distance between road segments and the connectivity of the road network.
- **Emission probabilities**: represent the likelihood of recording a given GPS position given an actual position on the road network. These probabilities can be based on the expected error in the GPS measurements.

The library can determine the most likely path on the road network given the observed GPS positions by applying the Viterbi algorithm [35], a dynamic programming algorithm used to find the most likely sequence of hidden states.

Trajectory Modeling

Trajectory modeling serves as a crucial building block for various smart mobility-related initiatives. These initiatives often rely on predicting future behavior based on past movement patterns, understanding the dynamics of the motion, and providing context to the movement. At its core, trajectory modeling refers to creating a model that accurately represents the movement of an object. This involves generating a mathematical or computational representation of the path that an object, such as a vehicle or a person, has taken or is predicted to take in the future. Given these two definitions:

Definition 3. (Road Network) A road network is a directed graph $G = (V, E)$, where V is a set of vertices v representing the intersections and terminal points of the road

[2] http://project-osrm.org/

[3] https://www.graphhopper.com/open-source/

segments, E is a set of edges r representing road segments, and each vertex has a unique id allocated from 1 to $|V|$. Moreover, we can consider each edge $r \in E$ corresponds to a road segment from a vertex $v \in V$ to another vertex $v'(v' \neq v) \in V$, where $r.s = v$ / $r.e = v'$ represent the start/end of the edge. [36]

Definition 4. (Mapped Trajectory) Given a raw trajectory T and a road network G, we map T to a road network path P which is composed of a set of connected edges in G, such that $T : r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_k$, and denoted as T containing a series of edge which captures the movement of an object from r_1 to r_2 and so on to r_k along the road network G, where every two consecutive road segments are connected, i.e., $\forall r_i, r_{i+1} \in T, r_i, r_{i+1} \in E \wedge r_i.e = r_{i+1}.s$.[36]

Trajectory modeling aims to model the likelihood of a given trajectory with length k, i.e.,

$$P(T) = P(r_1) \cdot P(r_2|r_1) \cdot P(r_3|r_1, r_2) \cdot \ldots \cdot P(r_k|r_1, r_2, \ldots, r_{k-1}) \tag{2.1}$$

2.6 Conclusion

In conclusion, Chapter 2 serves as a fundamental stepping stone into ethical AI practices and the intricate realm of trajectory data analysis. The exploration of Explainable AI (XAI) methods sheds light on their significance in enhancing model interpretability and transparency. Additionally, our journey through trajectory data's spatial and temporal intricacies underscores the critical role of data preprocessing techniques, with particular emphasis on map-matching. These techniques pave the way for our upcoming prediction tasks, which entail forecasting the final destination of trajectory data and predicting destinations moments before arrival. Furthermore, building upon the insights from XAI, we are primed to craft an interpretable white-box model for our forthcoming prediction tasks **"Explainable Trajectory Prediction"**, ensuring that the predictive power of AI is seamlessly intertwined with transparency and comprehensibility. This amalgamation of ethical AI practices, trajectory data understanding, and model interpretability forms a robust foundation as we journey toward our predictive endeavors, where AI-driven insights empower decision-making across various realms of our ever-evolving world.

Chapter 3

Trajectory Prediction

In the third chapter of this thesis, we focus on **Trajectory Prediction**, which forms the core of our research endeavor. The main objectives for this project involved not only predicting the ultimate destination of a moving object but also estimating its destination a few minutes before reaching the final point. These objectives are addressed using some appropriate approach considering the sequential nature of trajectory data.

The first part of the methodological approach is inspired by linguistic analysis techniques from the NLP field. Within this context, trajectories are interpreted as sequences of words or phrases, and each spatial location or zone is considered a discrete unit within the sequence - akin to a word or a phrase within a sentence. This analogy to language allows using established models like n-grams [37] for prediction tasks. The n-grams model, effectively predicts the next term in a sequence based on previous terms. In the context of trajectories, this equates to predicting the following location or spatial point based on historical trajectory data.

However, while n-grams models can provide compelling short-term predictions, they may not fully capture the complexity and dynamics of spatial trajectories. We have to note here that our 3-gram model achieved an accuracy of **71.56%** on the test data for predicting the last destination, a promising result that will be explored further in subsequent sections of this chapter. Nonetheless, the inherent limitations of n-gram model prompt us to seek a more advanced approach, which brings us to the application of Long Short-Term Memory (LSTM) [38] models. These recurrent neural network variants are ideally equipped to handle the non-linear relationships and temporal dependencies within the trajectory data, therefore holding great promise for our objective of accurately predicting trajectories. As we proceed in this chapter, we will delve deeper into applying LSTM models for our specific task.

LSTM is a particular Recurrent Neural Network (RNN) architecture specifically designed to handle sequential data, making it particularly appropriate for trajectory prediction tasks. Unlike traditional RNNs, which can suffer from the vanishing gradient problem during training, LSTMs incorporate a gating mechanism to control the information flow through the network. This allows them to effectively learn long-term dependencies in data, thus making them ideal for trajectory prediction tasks that necessitate a memory of past locations to predict future positions.

Utilizing Long Short-Term Memory (LSTM) models, we gain the capability to train on sequences of trajectory data. This then equips us to predict the destination, enhancing the predictive aspects of our task. The flexibility inherent in the architecture of LSTM models allows us to experiment with various configurations, even introducing dropout layers as a regularization technique to prevent overfitting. By fine-tuning our approach to meet the unique demands of our specific task, we have achieved a robust accuracy of **79.8%** on our test data using our LSTM model for predicting the last destination task.

While we initially concentrated on predicting a trajectory's final destination, we expanded our horizons to anticipate destinations just moments before arrival. The explanation of our LSTM model elucidated in the forthcoming chapter, illuminated a way forward. We understood, our model was trained to predict the last destination, and for improving it, we tailored our LSTM training process, focusing on fixed-sized sub-sequences of our trajectories. This enabled the model to decipher complex dependencies and recurring patterns within the data. By adopting this training approach, our model demonstrated an impressive capability to predict a trajectory's endpoint well before its actual arrival. Notably, when tasked with predicting a destination 2.5 minutes in advance, our model showcased a commendable accuracy of **47.92%**. This suggests that as a vehicle is close to its destination, our model can predict its final stop with nearly **48%** precision 2.5 minutes before.

Chapter 3 of this thesis unfolds a comprehensive exploration of trajectory prediction methodologies. Beginning with a thorough overview of the server system specifications and used dataset, the chapter delves into the heart of our strategies. It introduces the baseline n-gram model, detailing its application for predicting the last destinations and destinations moments before arrival. The chapter then navigates through LSTM Trajectory Prediction, discussing the LSTM architecture in detail and adapting it to predict the trajectory tasks. Concluding with assessing trajectory prediction approaches, this chapter sets the stage for subsequent discussions on specific methodologies and outcomes.

3.1 Server system specifications for this study

For this study, the Dragon2 server system at UMons was employed. This high-performance cluster features a SkyLake CPU with a clock speed of 2.60 GHz and 592 CPUs distributed across 17 nodes, each with 32 CPUs, and an additional 2 nodes with 24 CPUs. The system has substantial RAM ranging between 192 GB and 384 GB per node, facilitating resource-intensive tasks. Thanks to the 'Consortium des Équipements de Calcul Intensif' (CÉCI)[1] for allowing me to utilize the power of their server for my tasks.

[1] https://www.ceci-hpc.be/

3.2 Dataset

In this study, we have employed real-world GPS trajectory datasets named **Taxi Trajectory** [39] from taxis operating in Porto in Portugal. The size of this dataset is substantial, with a total volume of 1.8 GB, providing rich and diverse trajectory information for analysis. It comprises data generated by 442 taxis over an extensive period from January 7, 2013, to June 30, 2014. It contains a list of GPS coordinates, and they update every 15 seconds of the trip. The total travel time of a journey is defined as the (number of points-1) x 15 seconds. For example, a trip with 101 data points has a length of (101-1) * 15 = 1500 seconds.

3.2.1 Pre-processing of Data

As explained in the previous chapter, pre-processing should be applied before applying any model to data. In the initial stages of our data pre-processing, we explored the use of **clustering**, **rounding**, and **Map-matching** methods to reduce the dimensionality of the trajectory data and remove the noises.

For clustering, as a common technique in data science, we use the **K-means** [40] method to group similar data points together, in this case, similar trajectories based on specific characteristics. Rounding, another dimensionality reduction technique, was also experimented with. This method involves approximating trajectory data by rounding to a predefined level of precision. However, while these methods did manage to reduce the complexity of our data, they also led to a loss of important information, impacting the accuracy of our predictive models. Below, some drawbacks of using these methods are mentioned:

- **Spatial Resolution Loss**: rounding the trajectory data caused a loss in the spatial resolution of the data. When we used n-gram for predicting the destination, our model's next predicted points were common in most cases, and the model used popular paths for predicting the destination.
- **No Consideration of Road Network Constraints**: Using a k-means classifier doesn't consider the actual constraints of the physical road network. The predictions might be illogical or physically impossible in the real-world context (e.g., driving through buildings or bodies of water).
- **Inefficiency with Larger Datasets**: The use of the k-means classifier can become inefficient when dealing with a large number of data points. As the volume of data increases, the time complexity of the k-means algorithm also increases, potentially making it impractical for large datasets.
- **Insufficient Accuracy**: The obtained accuracy of the model could not meet our expectations.

Given these issues, we decided to employ the **map-matching preprocessing** method, which projects raw trajectory data onto an actual road network path. This method reduced the complexity of our data and preserved the significant information in the raw trajectory data. By enhancing the trajectory data with contextual road network information, we were able to achieve more nuanced analyses, which ultimately resulted in an improvement in the accuracy of our predictive models.

We employ **Leuven.MapMatching** as our map-matching algorithm based on the hidden Markov model to generate trajectories from raw GPS sequences. This map-matching

method is then used to project these sequences onto the road networks obtained from OpenStreetMap [41], focusing on extracting passenger-occupied trips. The steps of map-matching are mentioned following:

1. The road network data from our interested geographic area, Porto, Portugal is obtained by OpenStreetMap. Fig. 3.1 visualize the road network with the intersections and other important road segments as red points on a map.

2. Utilizing this information, a network (graph) representing the various segments of the road map was constructed, considering only roads traversed by a car.

3. After creating the graph, we aligned GPS points from the taxi trajectory dataset with the corresponding real-world road network. This was accomplished using a set of default parameters. It's important to note that the applied map-matching technic is based on the Hidden Markov Model (HMM). Within map-matching, the HMM is used to determine the most probable sequence of states — in this case, states matched on the road network — based on a series of observed points, which in our case are the GPS data. The rationale behind leveraging an HMM for map-matching is to accommodate the inherent uncertainty and noise in both the observed points and the road network. The HMM incorporates spatial relationships and the interconnectedness of the road network to estimate the most plausible sequence of matched points. It's noteworthy to mention that after applying the map-matching process to the original dataset of 1.7 million records, the size was reduced to 1.4 million. This suggests that our map-matching method was able to effectively match approximately 83% of the raw data, a significant portion. This demonstrates the effectiveness of our map-matching algorithm in correlating raw data points with their corresponding mapped locations.

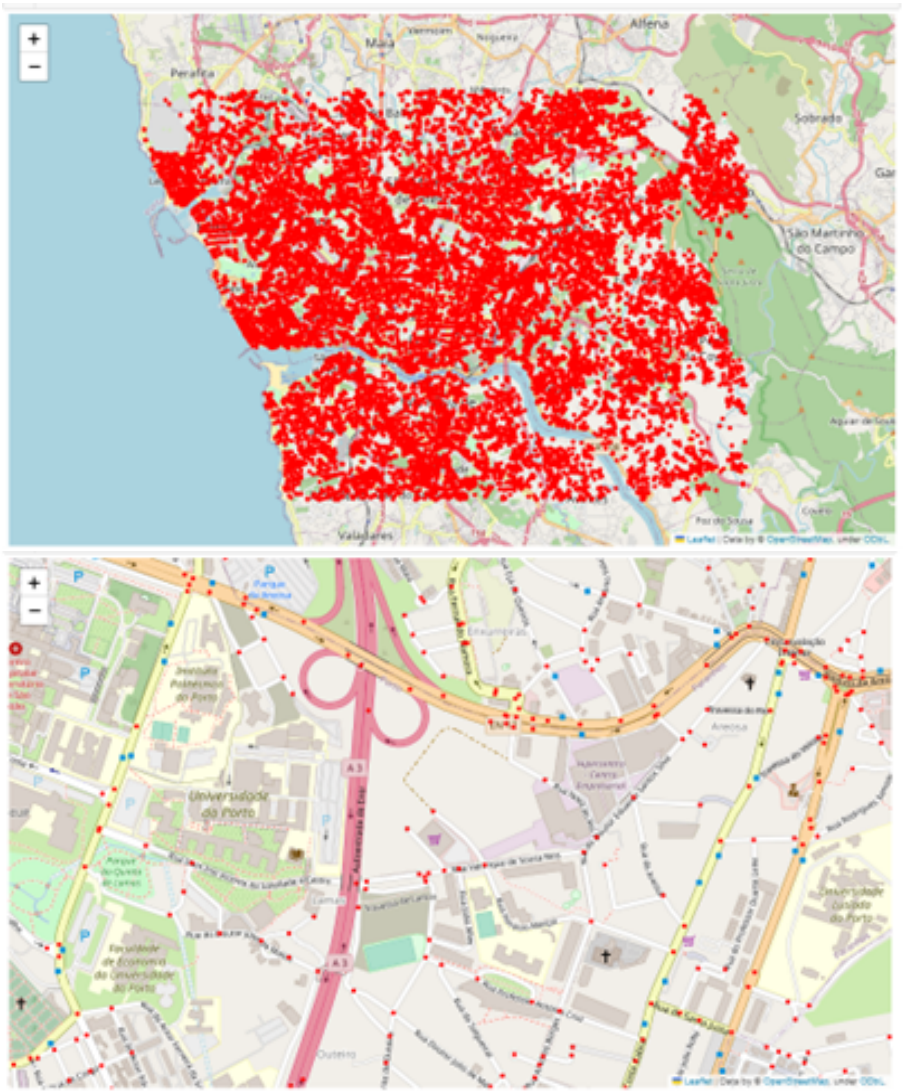

Figure 3.1: Obtaining the road network of Porto from OpenStreetMap. The upper map shows the whole road network in Porto and the lower map is one piece of the main map which is zoomed-in. The red points represent the road segments like intersections, three ways and other important road elements.

Figure 3.2 provides a two-part comparison of our map-matching samples. The top section presents a correctly map-matched sample, demonstrating the effectiveness of our method. In addition, the lower section portrays an incomplete instance of map-matching, providing further insight into the diversity of cases encountered in our dataset.

However, it is essential to note that the map-matching process may not always result in

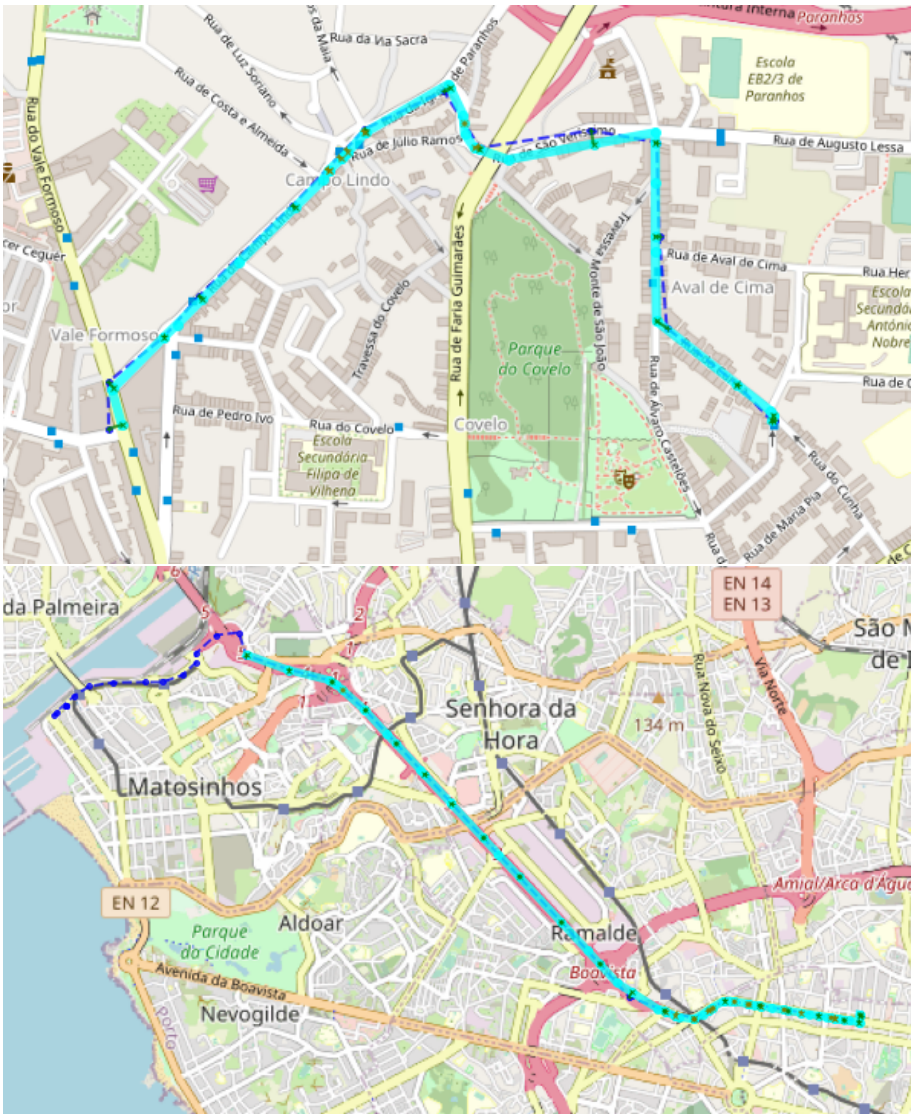

Figure 3.2: Visual comparison of different map-matching outcomes. The upper segment illustrates a successfully matched sample, while the lower section represents a distinct instance from the dataset which is not matched perfectly. Dark blue dashed lines represent original GPS trajectories, with the corresponding map-matched routes highlighted in illuminated blue.

perfect matches. For instance, the trajectory was not perfectly map-matched in the lower sample depicted in 3.2. Despite this, we included the mapped trajectory in our results because a road on the map aligns with our original GPS-recorded trajectory. Although

it may not represent an absolute match, this data could still provide valuable insights, especially in terms of the overall traffic flow and possible travel routes.

3.3 Baseline (n-gram model)

In natural language processing, n-grams are a widely adopted model that represents sequences of 'n' consecutive items commonly used in text processing. In our trajectory prediction task, an n-gram model allows us to represent trajectories as sequences of 'n' consecutive locations, enabling us to leverage the sequential nature of our data.

Given a mapped trajectory T on a road network G, we can define an n-gram model of size k in the context of these trajectories. Formally, a trajectory T is a sequence of road segments in G:

$$T = [r_1, r_2, ..., r_k] \tag{3.1}$$

An n-gram of size k from the sequence T is defined as the set of all k-tuples such that:

$$N\text{-}gram_k(T) = \{(r_i, r_{i+1}, ..., r_{i+k-1}) : \forall i \in \{1, ..., n-k+1\}\} \tag{3.2}$$

This set represents all possible sub-sequences of length k from the original trajectory T.

The following conditional probability can represent the predictive capability of the model:

$$P(r_{i+k}|r_i, r_{i+1}, ..., r_{i+k-1}) \tag{3.3}$$

The n-gram model uses the Markov assumption: This expression estimates the probability of the next road segment r_{i+k} given the previous k segments. These probabilities can then be used to model the likelihood of the entire trajectory T using the equation:

$$P(T) = P(r_1) \cdot P(r_2|r_1) \cdot P(r_3|r_1, r_2) \cdot ... \cdot P(r_k|r_1, r_2, ..., r_{k-1}) \tag{3.4}$$

Where each term $P(r_i|r_1, r_2, ..., r_{i-1})$ can be approximated using the frequencies of the n-gram in the training data. This expression estimates the probability of the next location l_{i+k} given the previous k locations which helps us to predict the last destination.

3.3.1 Predicting the last destination by n-gram

In our study, we used an n-gram model to predict the final destination of a trajectory on a road network. We investigated the performance of the n-gram model for various values of 'n' from 2 to 7. The results showed that the **3-gram** model yielded the highest prediction accuracy at approximately **71.56%**, followed by the **2-gram** model with an accuracy of roughly **69.48%**. The prediction accuracy gradually decreased as we increased the value of 'n' beyond 3. The **4-gram** model achieved an accuracy of **68.99%**, the **5-gram** model **65.97%**, the **6-gram** model **64.57%**, and the **7-gram** model **62.87%**.
This indicates that using the three most recent locations is the most effective for our trajectory prediction task according to our dataset, and adding more past locations into consideration does not improve the prediction accuracy but somewhat decreases it.

3.3.2 Predicting the last destination by n-gram some minutes before arrival

In this study, we refined our approach to focus on predicting the final destination just moments before arrival (from 6.25 to 2.5 minutes before). We employed an n-gram model to predict the endpoint of some points before reaching the destination. For instance, when we mention '2.5 minutes before', it implies that the n-gram model begins predictions of 10 road segments before reaching the destination. Likewise, '6.25 minutes before' indicates the model initiates predictions 25 road segments before the journey's end.
Suppose we have a 30-segment trajectory, and we are using a 2-gram model. If we aim to predict the destination 2.5 minutes before arrival, we can start using the trajectory's 21st point (30-10+1=21). Subsequently, we use this point to predict the 22nd point, the 22nd to predict the 23rd, and so on until we predict the final point. This method allows us to make sequential predictions within a given time frame.

The conversion of trajectory points to time is based on our dataset's specific characteristics: a new set of coordinates is updated every 15 seconds, meaning four points equate to one minute or $T = \frac{p}{4}$. During map-matching, multiple initial points may condense into a single road segment. Therefore, after experimenting, we found that paths with 20 trajectory points typically result in approximately 16 road segments post map-matching. We understand that the length of a trajectory before map-matching is either equal to or greater than the trajectory after the map-matching process $|P| > |T|$.

So, considering the final 10 points (or road segments) before arrival translates to a time interval of at least 2.5 minutes before reaching the destination.

Our results in 3.3 indicated that larger n-gram sizes increase prediction accuracy. For instance, a 6-gram model predicting the destination 2.5 and 6.25 minutes before arrival achieved accuracies of approximately 0.43 and 0.15, respectively, outperforming other models.

An interesting observation can be drawn when comparing the trends observed in different contexts of our study. As mentioned previously, we found that the overall prediction accuracy tended to decrease when we increased the value of 'n' beyond 3 in the previous task by the n-gram model. However, our results in Figure 3.3 demonstrated that specifically when predicting the final destination just a few minutes before arrival, larger n-gram sizes increased prediction accuracy. For instance, a 6-gram model predicting the destination 2.5 and 6.25 minutes before arrival achieved accuracies of approximately 0.43 and 0.15, respectively, outperforming other models.

The reason for this apparent contradiction lies in the specific way n-grams are used in the latter situation. When using a 6-gram to predict the last destination 10 points before (2.5 minutes before), we utilize the first 6 points to predict the following points. This context-specific use of the model allows for a more detailed and immediate mapping of recent trajectory points, thereby enhancing the model's predictive power for imminent destinations. Therefore, increasing the 'n' value can improve prediction accuracy in situations where recent trajectory data highly indicates the final destination.

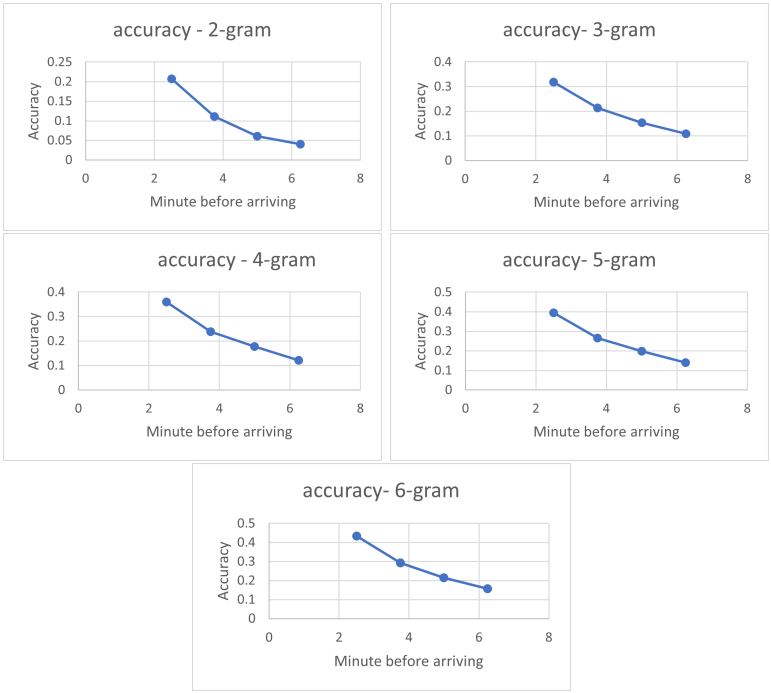

Figure 3.3: The result of using n-gram for predicting the final destination some minutes before arrival.

3.4 Trajectory Prediction by LSTM

After implementing n-gram as a baseline, we decided to go further and provide a more complex model for predicting the trajectories. For this purpose, we applied a Long Short-Term Memory (LSTM) network as a recurrent neural network (RNN) that excels in handling and predicting data sequences. Given the nature of inherently sequential trajectories, LSTM models are a logical choice. The primary objective is to use the preceding locations in a trajectory to predict the last location. This architecture offers a deeper understanding and ability to remember patterns over longer sequences, an advantage over simpler models like the n-gram which predicts based on a fixed preceding set of points.

Moreover, we focus on two distinct prediction tasks. The first task aims to **predict the final destination**, while the second **anticipates the destination a few minutes before its arrival**. We trained our LSTM model differently for each prediction task based on what we aimed to predict. In the subsequent sections, we delve into the specific details of each approach.

In the introductory chapter, we discussed the nature of RNN models, classifying them as "black-box" due to the challenges inherent in interpreting their inner workings. To strike a balance between performance and interpretability, especially for the subsequent stages of model explanation, we employed a streamlined LSTM architecture.

3.4.1 Architecture of LSTM model

There are different model architectures of RNNs that cater to a variety of sequence processing tasks. **One-to-One**, **One-to-Many**, **Many-to-One** and **Many-to-Many**. In our case, the model takes in a sequence of trajectory points and produces a single prediction. So, we can categorize our model in **Many-to-One** group where our prediction is the last position. Fig. 3.4 shows this architecture. (This diagram and some of the following architecture diagrams are adapted from Andrew Ng. Deep Learning Specialization course) [2]

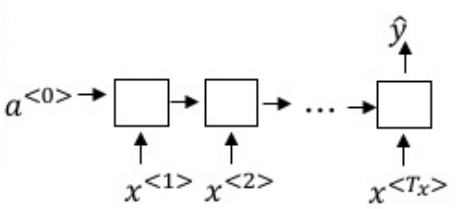

Figure 3.4: LSTM model architecture schema based on our task (Many-to-One)

Fig. 3.5 illustrates our model architecture in more detail. The process starts with **step 1**, an embedding layer that transforms trajectory points into dense vectors, thereby encapsulating the relationships between various points. Instead of representing each point as a unique and separate entity, embeddings map them into a shared space where similar points are closer together. This compact representation is more efficient than methods like one-hot encoding, which can be cumbersome for large datasets like our case study.

Following the embedding, the sequences, which have been padded to a uniform maximum trajectory length for efficient processing, feed into an LSTM layer. This padding is essential for batch processing in neural network training. When processing data in batches, every sequence in the batch must have the same length. Without uniform sequence lengths, batching would be impractical as matrix operations require consistent dimensions. By padding the sequences to a uniform length, we can harness the computational benefits of parallel processing inherent in batch training. This accelerates the training process and optimizes memory usage, especially when dealing large datasets. Padding trajectories to a uniform length ensure computational efficiency during model training and inference [42]. We first identified the sequence with the maximum length to standardize the sequence for batch processing. Subsequently, using a pre-padding approach, we appended zeros to the beginning of shorter sequences, ensuring uniformity across the dataset.

In fig.3.5, **Step 2** showcases the LSTM layer with 50 units. A model is considered deep when it has more than three layers, as illustrated in fig.3.6. They are crucial for understanding the sequential dependencies in trajectories. The layer uses a 'tanh' activation function to capture the temporal nuances of the trajectory sequences. The concluding dense layer, equipped with a softmax activation, provides a probabilistic forecast for the next location, treating the task as a multi-class classification.

[2] Andrew Ng. 2020. Deep Learning Specialization. Available at https://www.coursera.org

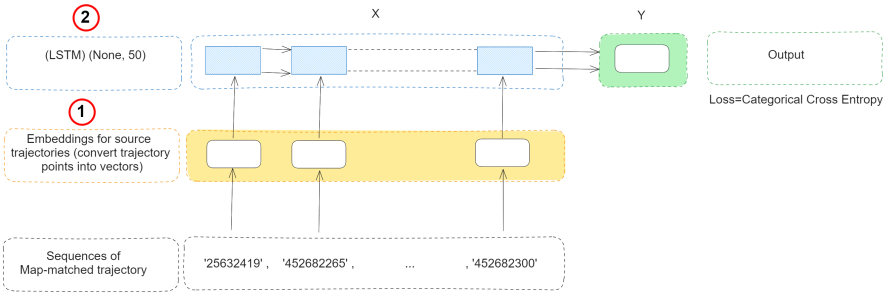

Figure 3.5: LSTM model prediction architecture schema.

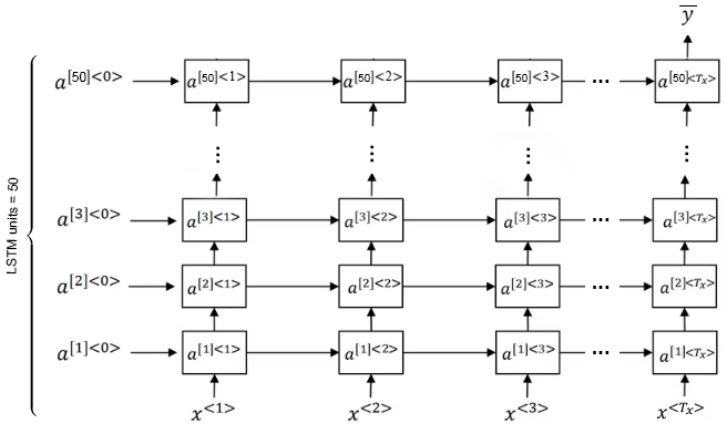

Figure 3.6: Schematic representation of our LSTM model architecture with 50 units. In this diagram, $X^{\langle 1 \rangle}$ to $X^{\langle T_x \rangle}$ represent the trajectory points in the input sequence.

To understand how each LSTM unit operates, it's crucial to delve into its core components and their interactions. The LSTM is explicitly engineered to tackle the challenges posed by long-term dependencies. Each unit's architecture is depicted in Fig. 3.7. In this figure, $x^{\langle t \rangle}$ represents the input at a specific time step t. The term $a^{\langle t-1 \rangle}$ signifies the activation from the previous time step, serving as a conduit for relaying past information, while $a^{\langle t \rangle}$ denotes the current activation at time step t. Similarly, $c^{\langle t-1 \rangle}$ encapsulates the cell state from the preceding time step, retaining crucial details about the sequence up to that juncture, and $c^{\langle t \rangle}$ represents the cell state at time t. Furthermore, each LSTM unit incorporates four distinct gates: the input(update) gate, the forget gate, the output gate, and the cell update gate. The subsequent equations elucidate their respective operations.

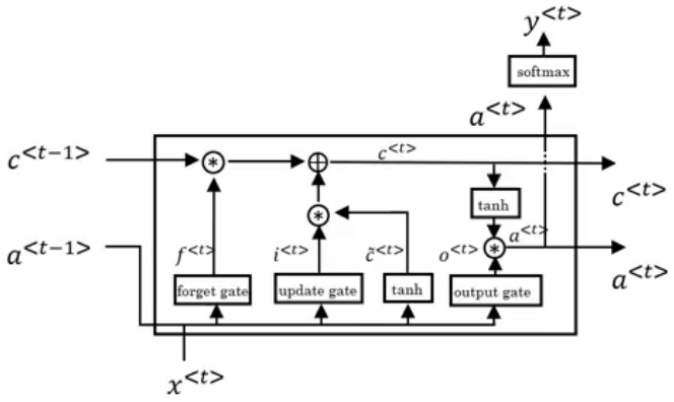

Figure 3.7: LSTM unit architecture schema.

$$\text{Forget Gate:} \quad f^{[\langle t \rangle]} = \sigma(W_f[a^{[\langle t-1 \rangle]}, x^{[\langle t \rangle]}] + b_f) \tag{3.5}$$

$$\text{Input Gate:} \quad i^{[\langle t \rangle]} = \sigma(W_i[a^{[\langle t-1 \rangle]}, x^{[\langle t \rangle]}] + b_i) \tag{3.6}$$

$$\tilde{c}^{[\langle t \rangle]} = \tanh(W_c[a^{[\langle t-1 \rangle]}, x^{[\langle t \rangle]}] + b_c) \tag{3.7}$$

$$\text{Update Cell:} \quad c^{[\langle t \rangle]} = f^{[\langle t \rangle]} \odot c^{[\langle t-1 \rangle]} + i^{[\langle t \rangle]} \odot \tilde{c}^{[\langle t \rangle]} \tag{3.8}$$

$$\text{Output Gate:} \quad o^{[\langle t \rangle]} = \sigma(W_o[a^{[\langle t-1 \rangle]}, x^{[\langle t \rangle]}] + b_o) \tag{3.9}$$

$$a^{[\langle t \rangle]} = o^{[\langle t \rangle]} \odot \tanh(c^{[\langle t \rangle]}) \tag{3.10}$$

Where:

- $x^{[\langle t \rangle]}$ is the input at time step t.
- $a^{[\langle t-1 \rangle]}$ is the activation from the previous time step.
- W and b are the weight matrices and bias vectors for each gate.
- $\odot$ denotes element-wise multiplication.
- $c^{[\langle t \rangle]}$ is the cell state at time step t.

3.4.2 Predicting the last destination

When training our LSTM model for this task, we treated the entirety of a trajectory, excluding its last point, as the input data (denoted as X). The final point of each trajectory was set as the target output (denoted as y). This configuration allowed the model to learn from the sequence of trajectory points and predict the last point based on that sequence.

After training the LSTM model, we proceed to the testing phase. A new set of trajectories (excluding their last point) is fed into the trained model for prediction. The model then outputs a predicted point for each trajectory, which we expect to align closely with their actual last point. By comparing the model's predictions with the real last points of the trajectories in the test set, we can assess the accuracy and performance of our model.

It should be mentioned that our dataset was divided into three distinct portions: 70% for training, 20% reserved for validation, and 10% for the testing process.

The LSTM model achieves training accuracy **80.95%** and validation accuracy **80.86%** and test accuracy of **79.8%**. It signifies that it correctly predicts the final destination in a trajectory **79.8%** of the time based on the preceding locations. This accuracy level suggests underlying patterns in the trajectory sequences the model has successfully captured. While the architecture isn't overly complex, it balances model complexity and interpretability. Moreover, compared to the n-gram approach, LSTMs can work with varying lengths of input sequences, giving them flexibility in learning from short and long trajectories.

3.4.3 Predicting the last destination some minutes before arrival

After some experiments especially explaining the model (More information is available in the next chapter), we understood that our model should be trained differently to predict the last destination some minutes before arrival. In other words, our model learned to focus on some prior points to predict the last destination. So, instead of training on full trajectories, we employ a sliding window technique to train the model for this task. This involves using fixed-size windows that move through the trajectory, transforming each trajectory into sub-sequences of the main trajectory with fixed length. This methodology is particularly adept at predicting the next positions in a trajectory and is also well-suited for tasks where the goal is to predict the last destination a few minutes in advance. The emphasis here isn't on altering the LSTM's architecture as mentioned in Section 3.4.1, but on refining how the data is presented to it, enhancing its prediction capabilities for the given task.

The strength of this approach is data utilization. Unlike the previous training method that treats each trajectory sequence as a single data for predicting the final destination, this method maximizes data utilization by considering the entire length of trajectories. The model effectively captures the intricate dependencies and patterns within the data by transforming trajectory sequences into fixed-size sequences. Fig. 3.8 shows the differences between the number of sequences made by each trajectory in both training methods. If we consider the trajectory length is 11, and the fixed size sequence length is 6, we will have five sub-sequences instead of one sequence.

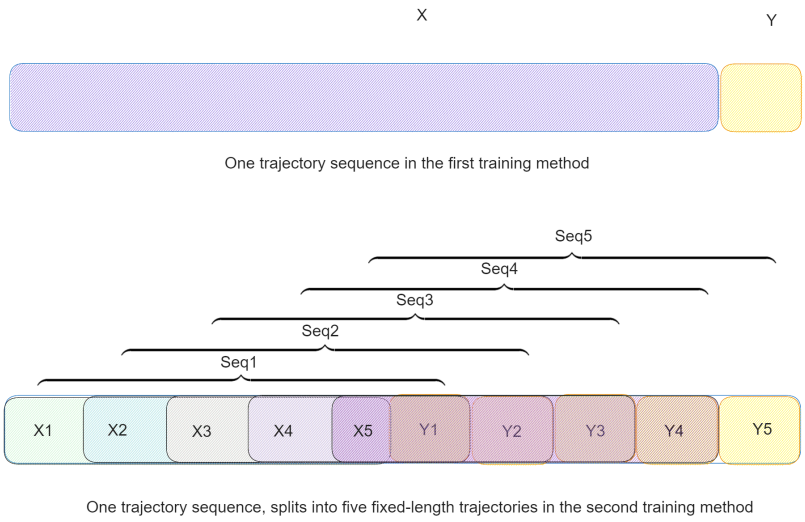

Figure 3.8: The differences between the number of sequences in both training methods

Optimal windows size

To determine the optimal window size, we referred to the insights garnered from our LSTM model explanation in the previous task, which indicated that a size of 6 was the most effective—with 5 points designated for x and 1 point for y. Additionally, the n-gram model result in predicting the last destination a few points earlier revealed that the 6-gram model yielded the highest accuracy. Based on these findings, we settled on a window size of 6.

It should be noted that an RNN could be employed in place of the LSTM for this approach. Given that each input has a fixed size of 6, the advanced memory capabilities of the LSTM might not be essential. Opting for the RNN could enhance efficiency by reducing model complexity. However, we retained the same LSTM architecture to maintain uniformity across both tasks.

The result of prediction

Before we delve into the results, let's clarify our prediction process and mention that for this task, we splited the data into three portions: 60% for training, 20% reserved for validation, and 20% for the test in the context of the last destination prediction some minutes before arrival.

We use a specific strategy for our test samples to anticipate the final destination a few minutes before arrival. Here's how we proceed when aiming to predict the endpoint for instance 10 steps (or 2.5 minutes) in advance:

- **Step 1:** For a given trajectory T, we partition T into two components: x and y. Here, y represents the ultimate destination, while x encompasses all the ordered trajectories leading up to y.

- **Step 2:** Depending on our predetermined number of endpoints (e.g., nine endpoints here as y was already truncated), we truncate the final ten data points from x. Mathematically, this can be represented as $x = x[:-9]$.
- **Step 3:** Leveraging the retained trajectory points in x and specifically the last 5 points within x, we forecast the subsequent destination at position $t + 1$. This predicted destination is then appended to x, facilitating the prediction of $t + 2$.
- **Step 4:** This iterative prediction is executed 10 times to forecast all requisite points. Ultimately, we compare the final predicted trajectory point with y. If they match, the prediction is deemed successful, indicating that the model has accurately traced the path to the authentic y.

In the context of this experimental setup, our model manifested the following prediction accuracies as well as the fig. 3.9:

- 6.25 minutes before arrival: The model achieved an accuracy of **18.16%**.
- 5 minutes before arrival: The prediction accuracy was found to be **23.95%**.
- 3.75 minutes before arrival: Our model showcased an accuracy rate of **32.54%**.
- 2.5 minutes before arrival: Impressively, the accuracy soared to **47.92%** as we neared the destination.

These results signify the model's capability to make sequential predictions within a specific time frame and underscore its increasing precision as the trajectory approaches its final point.

It should be noticed that the training and validation accuracy for this task were **92.83%** and **92.25%** respectively. Since this model is trained to predict the next position, it can also be used for predicting the last destination. In this situation, we need to input the five last point to predict the last point.

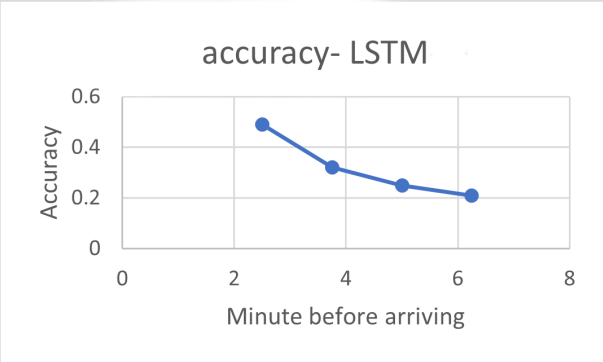

Figure 3.9: The accuracy of the LSTM model with fixed-windows sub-sequences training approach for predicting the last destination some minutes before arrival.

3.5 Conclusion of trajectory prediction approaches

In culmination, Chapter 3 offers an in-depth exploration of two distinct trajectory prediction tasks. To tackle these tasks, we employed two different models: the n-gram, serving as

our baseline, and the LSTM as a complex model. While we retained the LSTM structure for both tasks, the training methodologies varied.

For one of our tasks, we utilized a sliding window technique; we transformed entire trajectories into fixed-length sub-sequences, maximizing data utilization and capturing intricate dependencies within the trajectory data.

A comprehensive comparison of the outcomes from our varied methodologies, including their respective accuracies and performances, is presented in the 3.1 table.

Prediction Task: Last Destination		
Model	**Detail**	**Accuracy(%)**
n-gram	3-gram	71.56
	2-gram	69.48
	4-gram	68.99
	5-gram	65.97
	6-gram	64.57
	7-gram	62.87
LSTM	-	79.8
LSTM	trained with sub-sequences	**92.25**

Prediction Task: Last Destination Some Minutes Before Arrival		
Model	**Minutes before arrival**	**Accuracy(%)**
2-gram	6.25 mins	4.05
	5 mins	6.05
	3.75 mins	11.11
	2.5 mins	20.73
3-gram	6.25 mins	10.85
	5 mins	15.38
	3.75 mins	21.41
	2.5 mins	20.73
	6.25 mins	31.79
4-gram	6.25 mins	12.17
	5 mins	17.71
	3.75 mins	23.89
	2.5 mins	35.91
5-gram	6.25 mins	14.06
	5 mins	19.93
	3.75 mins	26.5
	2.5 mins	39.44
6-gram	6.25 mins	15.79
	5 mins	21.39
	3.75 mins	29.26
	2.5 mins	43.42
LSTM(trained with sub-sequences)	6.25 mins	**18.16**
	5 mins	**23.95**
	3.75 mins	**32.54**
	2.5 mins	**47.92**

Table 3.1: Comparison of Prediction Accuracies for Different Models and Tasks

Chapter 4

Explainable Trajectory Prediction

After the prediction phase, the models are analyzed and explained based on the theoretical principles and methodologies in the second chapter. This process stage is crucial for understanding the model's behavior, providing an interpretive layer over deployed models' mathematical and algorithmic intricacies. Through this, we can gain insight into the models' predictive abilities, identify potential shortcomings, and pave the way for future improvements.
Chapter 4 delves into the explainability of trajectory prediction models, focusing on the n-gram and LSTM approaches. It starts with the n-gram model explanation. Then, continues with the explanation methods for LSTM. Furthermore, we employed two distinct methods to explain the LSTM model and subsequently compared their results. This exploration enhances our comprehension of model behavior and drives the development of a solution for improved trajectory prediction.

4.1 Explanation of n-gram model

The n-gram model stands in stark contrast to many of the complex models discussed in Chapter 1 with respect to explainability. Its inherent simplicity and determinism offer it the unique distinction of being a *white-box* model. Unlike deep neural networks or ensemble methods, which often operate as *black-boxes*, the n-gram model bases its predictions on directly observable patterns and frequencies present in the historical data.

This directness in its operational mechanism negates the need for specialized explanation methods, as each prediction can be traced back to specific sequences present in the training data. Moreover, its reliance on the last $n-1$ data points ensures that predictions are contextual and localized, further simplifying the explanation process.

We will not resort to the explanation methodologies described in Chapter 1 to elucidate this model's transparency. Instead, we shall illuminate the workings of the n-gram model through practical examples, demonstrating its inherent explainability.

At its core, an n-gram model works on the principle of predicting the next item in a sequence based on the previous $n-1$ items. Indeed, the primary strength of the n-gram model in trajectory prediction lies in its ability to focus on the most recent data points

for prediction. While a trajectory might encompass a long sequence of road segments or points, for predicting the last destination, only the most recent $n-1$ points play a pivotal role, making prior segments or initial points of lesser consequence in the decision-making process.

4.1.1 Explanation of n-gram model for predicting the last destination

Imagine a long trajectory of road segments represented as $A, B, C, D, E, F, G, H, I, J, K, L, M$; we aim to predict the final destination, which in this case is M. Refer to fig. 4.1 for a visual representation of this trajectory. To utilize the 4-gram approach for this prediction, we consider the last 3 observed points in the trajectory and attempt to predict the next one. In our scenario, the last three points before M are J, K, and L. So, the 4-gram model will look up its training data for occurrences of sequences that start with J, K, and L to predict the most likely next segment.

If, during training, the sequence J, K, L, M was frequently observed, then the model would predict M as the next segment with high confidence. This prediction is purely based on the patterns and frequencies observed in the training data. The model essentially states: "Given that I've seen J, K, and L in sequence, the next most probable segment, based on my historical data, is M".

The strength of the n-gram model, especially in the context of trajectory prediction, lies in its simplicity and direct reliance on observed patterns. It doesn't account for the entire sequence but is heavily influenced by the most recent $n-1$ segments, making its predictions and rationale straightforward to understand and explain.

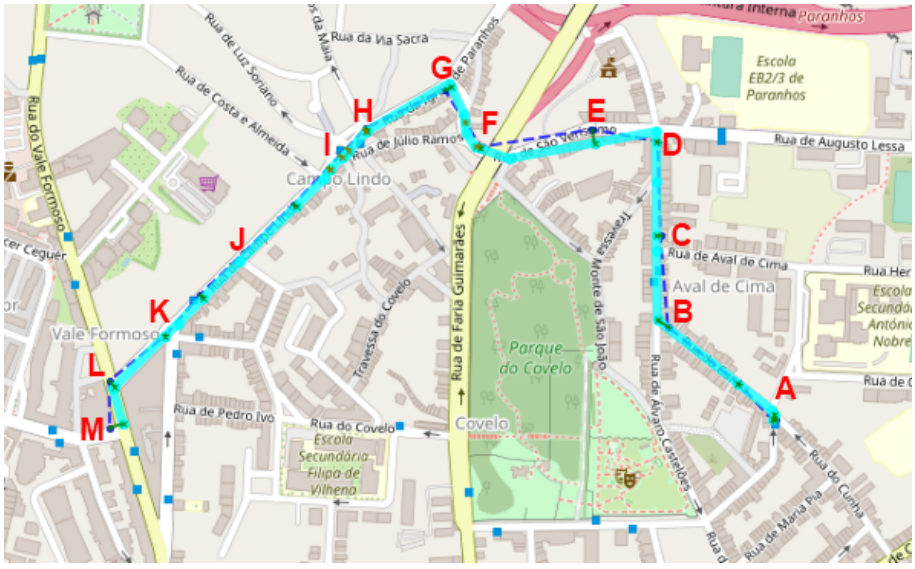

Figure 4.1: Example of trajectory prediction by the n-gram model approach

4.1.2 Explanation of n-gram model for predicting the last destination some minutes before arrival

Predicting the endpoint of a trajectory some minutes before arrival introduces a more intricate challenge. The model must anticipate the trajectory's course further down the line, based on the most recent segments and historical patterns.

Consider previous example trajectory 4.1 $A, B, C, D, E, F, G, H, I, J, K, L, M$. Utilizing a 4-gram model and aiming to predict the final destination (segment M) 5 segments (or 1.25 minutes) in advance would mean that upon the vehicle reaching segment H, the model would evaluate sequences in the format $E, F, G, ?$. Referring to past data, it will identify which segment typically follows the sequence E, F, G and predict point H, then predict point I, and so on. Finally, with predicting H, I, J, K, L correctly, the model can predict the last destination accordingly.

Through the n-gram approach in trajectory prediction, decisions are primarily rooted in recent segments or points. This ensures that predictions reflect the immediate context and are based on historically observed patterns.

From both the aforementioned examples, an insightful observation can be delineated: early segments or points in a trajectory are not imperative for the n-gram model when predicting the final destination or when making a prediction a few segments before the end. The very nature of the n-gram model allows it to focus solely on the recent context, i.e., the last $n-1$ segments. This is especially advantageous in real-world scenarios where only the latest data might be readily available or when the trajectory's early data is considered irrelevant or noisy. Hence, the model's capacity to ignore early segments and concentrate on the more recent ones fortifies its applicability and efficiency in trajectory prediction tasks.

4.2 Explanation of LSTM model

To explain our model, we deployed two methods: a variant of LIME (Local Interpretable Model-agnostic Explanations) which is **LIME Text Explainer** and another method that draws inspiration from the LIME framework, shaping our approach to shed light on the intricate behavior of our model. However, our method introduces certain distinctions to cater to the unique characteristics of our trajectory prediction task. Initially, we delineate the primary differences between the two models. Following this groundwork, we introduce our approach. Subsequently, we present the findings from the LIME Text Explainer and compare the results of both techniques.

LIME operates by generating perturbations of the input data, then evaluating how these perturbations affect the model's predictions, subsequently utilizing simpler, interpretable models to elucidate the predictions of the more intricate model. Drawing inspiration from this, our methodological approach involved the deliberate perturbation of trajectory data to discern its impact on the LSTM model's predictions. However, they have some fundamental differences such as:

- **Scope and Focus:** LIME is primarily designed to explain individual predictions locally. It perturbs and probes the input features of a single instance to understand how changes affect the model's output for that instance. Our approach has focused more on understanding the model's behavior at a global level, considering patterns and dependencies across multiple instances or sequences. This allows us to analyze broader trends in how perturbations impact predictions.

- **Interpretable Models:** LIME often uses interpretable surrogate models to approximate the behavior of the complex model locally for the specific instance being explained. Our approach doesn't involve using surrogate models explicitly; instead, it focuses on the direct impact of perturbations on the model's predictions.
- **Model Complexity:** LIME is a general framework that can be applied to many models, from simple linear models to complex deep neural networks. Our approach is focused on explaining the behavior of a specific model designed for trajectory prediction.
- **Data Type:** LIME is more commonly applied to tabular and structured data, where each feature is treated independently, without considering the sequence or temporal aspect. Our approach is tailored to sequential data, specifically trajectory data. This data type has inherent temporal dependencies and order, requiring specialized perturbation techniques considering the sequence structure.

Based on what we mentioned in the last item (Data type), our approach entailed the introduction of points from neighboring trajectories and removing various trajectory segments, particularly focusing on the model's reliance on the terminal points in a given sequence. The steps of work are mentioned in Section 4.2.1.

4.2.1 Explanation of LSTM model for predicting the last destination

Step 1: Identification of Similar Trajectories using Dynamic Time Warping (DTW):

Given the objective of introducing real points for trajectory perturbation, it was first essential to identify trajectories closely aligned with our case study trajectory. The Dynamic Time Warping (DTW) algorithm was employed to achieve this. DTW is renowned for its capability to measure similarities between two temporal sequences.

There are various methods for identifying the similar trajectories like **Edit distance** [43] and the **Dynamic Time Warping (DTW)** [1]. In this study, we employed DTW because we used trajectory data before map-matching which is inherently noisy. To counteract this noise, we opted for DTW because it calculates squared distances between all paired aligned points, providing tolerance to outliers. Thus, with the squared distance metric, DTW can effectively handle sporadic anomalies in trajectories, proving beneficial for data prone to irregularities or outliers [44]. Furthermore, after visualizing the outcomes of this step, we found that the algorithm met our criteria, prompting us to proceed with this method.

Step 2: Pairing with the Least Distance: Post the DTW computation, pairs of trajectories were ranked based on their DTW distances. The pair with the least distance indicated the closest match regarding spatial alignment and movement pattern. In fig. 4.2, on the left map, you can observe the case study trajectory (represented in red on the map) and its closest neighboring trajectory (shown in blue) were identified in this context.

Step 3: Selection of Points for Perturbation: With the neighboring trajectory (blue) identified, specific points or segments from this trajectory were chosen for perturbation purposes. The rationale behind this choice was to harness genuine trajectory data points, ensuring the perturbed trajectory remains grounded in actual movement patterns and spatial reality. The disturbed trajectory is shown in fig. 4.2, on the right map.

[1] Berndt and Clifford 1994

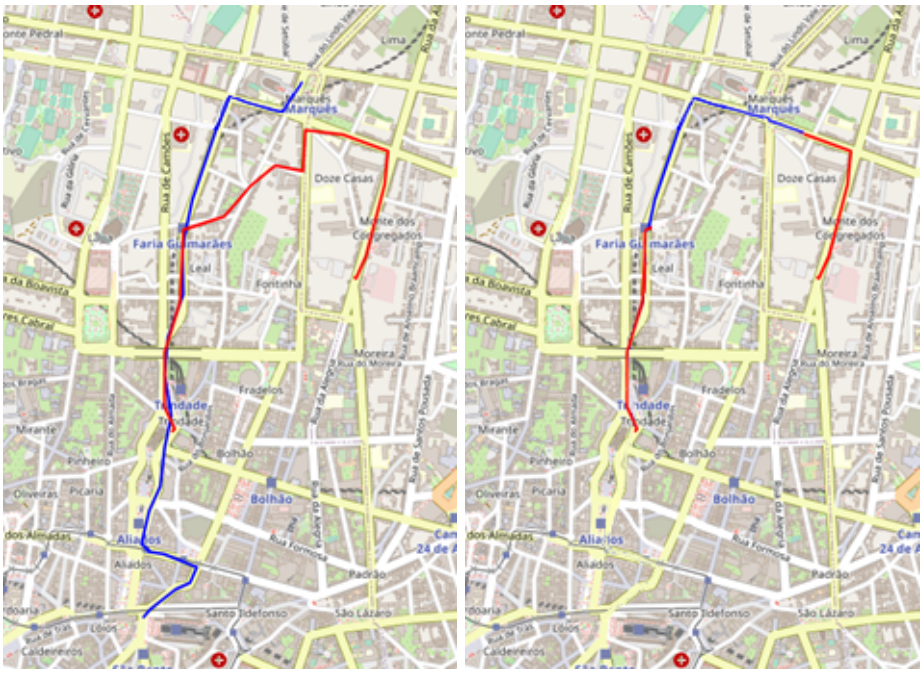

Figure 4.2: The steps of perturbing the case study trajectory by a neighbor trajectory

Step 4: Model Prediction Evaluation Following the trajectory perturbation, the original case study trajectory (red) and the perturbed one (infused with blue trajectory points) were fed into the LSTM model as the test set in predicting the last destination task. The subsequent predictions were then meticulously analyzed to discern the impact of the blue trajectory points on the predictions of the red trajectory. Notably, we observed that the final destination in both trajectories remained identical, suggesting that **perturbations in the middle of a trajectory had no tangible influence on the model's prediction outcome**. This exercise assessed the model's robustness to perturbations and provided a spatial and intuitive understanding of the model's behavior in the presence of disturbances.

Step 5: Further Model Behavior Analysis To delve deeper into the model's behavior, we performed additional tests by perturbing different sections of our trajectory: the beginning, middle, and concluding. Through systematic alterations and subsequent evaluations, a recurring pattern emerged. Only when the three last points of the trajectory were altered did the model's predictions deviate from the expected outcome. This observation highlighted an intriguing characteristic of our LSTM model: its behavior in predicting the last destination in trajectory data mirrors the properties of an n-gram model, emphasizing the significance of the immediate prior sequence for predictive outcomes.

Upon closer observation of our model's behavior with trajectories, we decided to refine our test dataset, which originally consisted of **285,043 trajectories**. Motivated by the insight that the model predominantly leveraged the immediate prior points for its predictions, we made a strategic decision to retain only the three last points of each trajectory for testing. The result was enlightening: the accuracy stood at **77.4%**. Further experiments, where we retained the trajectories' last four and five points, yielded accuracies of **78.5%**

and **78.9%**, respectively. These figures were compellingly close to the original test data's accuracy of **79.8%**. This evaluation underscores the power of concise data representation, especially when the model's inherent behavior emphasizes recent trajectory points.

Based on our examination of the LSTM model, the key probable factors influencing our observations is **Data Patterns**: Our analysis hints at the possible data structures within our trajectories. The decisive patterns that predominantly influence the final destination seem to be located within the concluding segments of each trajectory sequence. As a result, the LSTM model efficiently discerns these vital cues, emphasizing their role in its predictions.

4.2.2 Explanation of LSTM model with LIME Text Explainer

The "LIME Text Explainer" is a specific implementation of LIME designed to work with text data. Here's a breakdown of how it operates based on this website [45]: Upon receiving a sentence, we initially create a bag of words (BoW) representation for it. The library then selectively picks words from this initial sentence, rearranging them to produce 5000 varied sentences with different word sequences. These newly formed samples are assigned weights based on their similarity to the original sentence. LIME proceeds with its standard process, measured using the cosine distance with these weighted, vectorized sentence samples.

We used this approach to explain our model for predicting the last destination for the previous section sample, and the result is shown in fig.4.3.

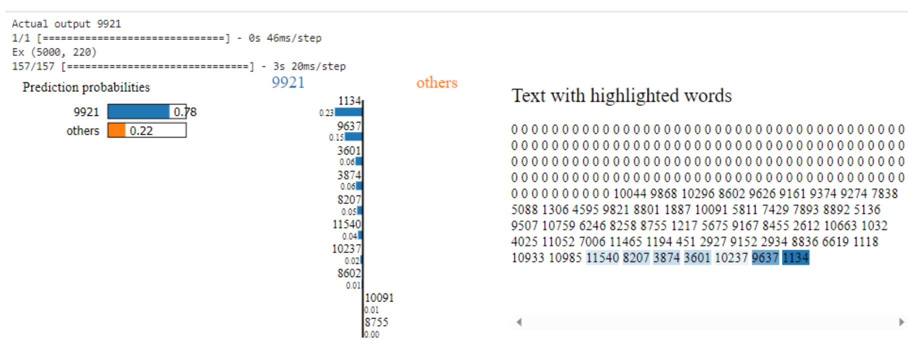

Figure 4.3: Using LIME Text Explainer to explain our LSTM prediction on one sample of trajectory

The explanation results show that the model has accurately predicted this sample. Both the actual output and the predicted point are denoted as 9921. The trajectory points marked in blue have played a pivotal role in steering the prediction toward point 9921, while those highlighted in orange influenced the predictions toward other points. It is observed that the last trajectory points play a significant role in influencing the prediction of point 9921 as the predicted destination, which is correct.

From the results, it's clear that our approach for explaining the model's decisions is accurate and reliable. Since it is evident that the concluding trajectory points are pivotal in forecasting the final destination, mirroring the insights gleaned from the LIME Text Explainer. This gives us confidence in the model's predictions and its overall functionality.

4.2.3 Explanation of LSTM model for predicting the last destination some minutes before arrival

From the detailed explanation of the LSTM model's behavior for predicting the last destination, it's clear that the model heavily relies on the immediate prior sequence for its predictions. This characteristic of the LSTM model aligns with the properties of an n-gram model, where predictions are driven by recent segments of the trajectory. However, it's important to highlight a key distinction between the two approaches: while both LSTM and n-gram models focus on utilizing historical information to make predictions, the n-gram model moves along the trajectory, capturing local dependencies between segments, whereas the LSTM model processes the entire trajectory as a sequence. This distinction becomes particularly relevant when attempting to forecast a destination some minutes before arrival. In this scenario, the LSTM's concentration on the immediate context might not encapsulate the full scope of the trajectory's future path, resulting in limited accuracy. This limitation arises from the learning method of LSTM. We decided to change the learning method to address this problem and empower the LSTM to predict the destination some minutes before. In this scenario, instead of training on full trajectories, we employ a sliding window technique to train the model, leading to the model explained in Section 3.4.3. As a recall, this involves using fixed-size windows that move through the trajectory, transforming each trajectory into sub-sequences of the main trajectory with fixed length.

To determine the optimal window size, we observed the impact of the final data points in both explanation methods. Through our analysis, a window size of 6 emerged as the best balance between accuracy and computational complexity. Notably, this size was recurrently observed as an average across all our examined samples.

Applying our explanation approach on LSTM model

To verify the validity of our explanation, we conducted a perturbation experiment using random trajectory points. By disturbing our case study trajectory with one, two, three, and four starting points, we precisely observed the model's behavior in predicting the last destination some minutes before arrival.

For example, consider one example of our case study trajectory:

$$[4229, 4990, 2573, 3312, 3138, 3189, 2113, 8339, 2056]$$

where 2105 is the actual last destination.

We aimed to show different scenarios for disturbing initial points of this trajectory.

- **One-point perturbation**:

  $$[9999, 4990, 2573, 3312, 3138]$$

  (disturbed point: 9999)

- **Two-point perturbation**:

  $$[1111, 2222, 2573, 3312, 3138]$$

  (disturbed points: 1111, 2222)

- **Three-point perturbation**:

  $$[4444, 5555, 6666, 3312, 3138]$$

  (disturbed points: 4444, 5555, 6666)

- **Four-point perturbation**:

$$[7777, 8888, 9999, 4444, 3138]$$

(disturbed points: 7777, 8888, 9999, 4444)

The graphical representation in Figure 4.4 vividly illustrates the relationship between disturbance of initial trajectory points and subsequent changes in accuracy. Evidently, perturbing the first point of the trajectory yields minimal impact on accuracy, akin to the precision achieved with the original trajectory. Upon disturbing the first two points, the accuracy remains commendable. However, a significant decline in accuracy is noticeable upon disturbing three points, where it diminishes to approximately **20%** across most experiments. Intriguingly, the accuracy plummets close to zero upon disturbing four points. This graphical depiction underscores the sensitivity of the model's predictions to disturbances in the initial points of the trajectory, reinforcing the importance of the preceding context for predicting the last destination some minutes before arrival. These findings echo our previous experiments with the LSTM model, where we discovered the remarkable accuracy achieved when predicting the next point with only three preceding points, further accentuating the impact of the immediate context in shaping accurate predictions.

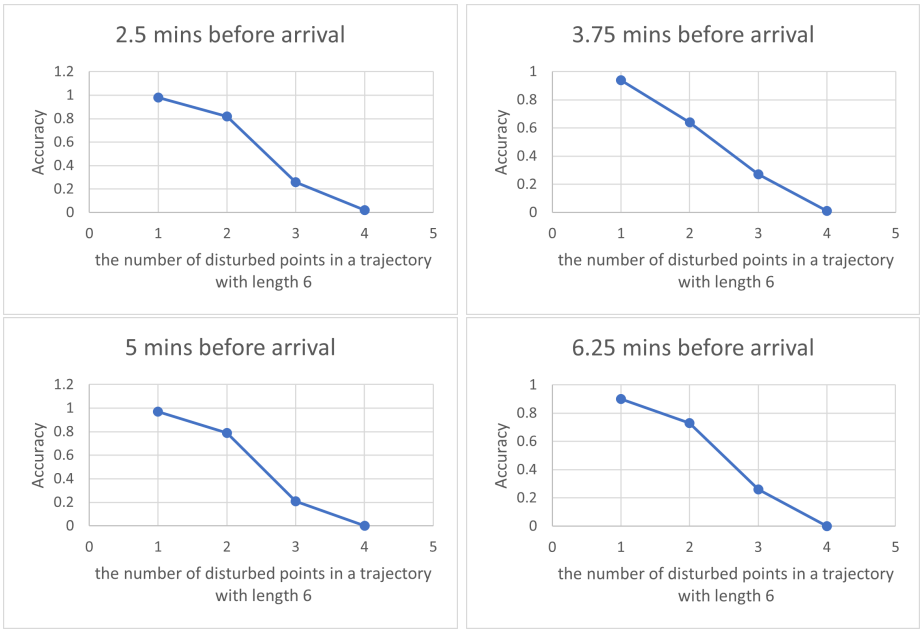

Figure 4.4: Explaination of the LSTM model with disturbing some points in trajectories with a length of 6 to predict the last destination some minutes before arrival.

It's important to note that in conducting this test, we first identified the accurate samples that could predict the last destination some minutes before arrival. Subsequently, we selectively perturbed the initial points of these validated samples to observe the resultant effects. Consequently, the reported accuracy figure corresponds to the percentage of

successful predictions of the last destination when using disturbed trajectories across all validated samples. To illustrate, consider a scenario with 1000 trajectory test data points. If the accuracy for predicting the last destination 2.5 minutes before arrival is recorded at 50%, it signifies that we would conduct this perturbation experiment on 500 validated samples. Furthermore, if the achieved accuracy in this task is 80%, it implies that out of these 500 samples, 400 samples were predicted accurately despite the disturbances introduced during the experimentation process.

Applying LIME Text Explainer on LSTM model

In our pursuit of model understanding, we employed the LIME Text Explainer on some predicted samples. Figure 4.5 showcases the explanatory outcome on two distinct samples — one predicted accurately and the other inaccurately. The likelihood of accurate prediction increases when the model emphasizes the concluding data points which aligns with what we have shown in the previous section. In the majority of scenarios, all five data points significantly influence the prediction. Yet, there are instances where emphasis is also placed on the latter points, as depicted in the upper figure. Conversely, when the model neglects the final data point, its prediction accuracy for the subsequent point tends to diminish as shown in the lower figure.

Figure 4.5: Using LIME Text Explainer to explain our LSTM prediction on two samples of trajectory with length 6; the upper prediction is correct and the lower is not correct

4.3 Conclusion

The exploration into trajectory prediction models, specifically the LSTM and n-gram approaches, has revealed nuanced insights into their behavior and potential synergies. In the endeavor to demystify the LSTM model's behavior for trajectory predictions, we employed LIME Text Explainer and perturbation tests. LIME Text Explainer provided invaluable insights into the word or, in this context, trajectory point importance. This method illumi-

nated the crucial nature of the concluding trajectory points, confirming their dominant role in shaping the model's prediction outcomes. Our perturbation experiments further underscored this discovery. By introducing disturbances at various points within the trajectory and monitoring the resultant prediction accuracy shifts, we were able to obtain a granular understanding of how the LSTM model perceived and prioritized different segments of a trajectory.

The decision to convert the training set into fixed-length subsequences arose from a keen observation of the LSTM model's behavior. Since the model demonstrated a pronounced emphasis on the immediate prior sequence, capturing the nuances of shorter sequences became imperative. Using a sliding window technique, each primary trajectory was transformed into shorter sub-sequences, allowing for a more focused and localized training approach. This ensured that the LSTM could better discern patterns within smaller trajectory segments and potentially increase its prediction accuracy, especially when forecasting a destination some minutes before arrival. By concentrating on subsequences, we aimed to harness the model's inherent proclivity towards recent data points and optimize its predictive prowess within that context.

Chapter 5

Conclusion

The first step for predicting the trajectory is data pre-processing. Map-matching as our pre-processing method converts raw GPS coordinates into trajectory sequences by aligning them to actual roads. For our case study which is Taxi Trajectory data from Porto, Portugal, we used the Leuven.MapMatching algorithm, based on the Hidden Markov Model, to associate GPS points from taxi data to OpenStreetMap roads. This algorithm adeptly handled GPS and road data uncertainties, effectively matching approximately 83% of the raw points to their true road positions. The process transformed 1.7 million records to 1.4 million, highlighting its precision.

In the process of trajectory prediction, our exploration predominantly revolved around two tasks: predicting the last destination and estimating it a few minutes before arrival. These two tasks are accomplished with two different approaches. The main approach is LSTM, and the baseline is n-gram.

In our research, we used n-gram models for trajectory predictions as the baseline on a road network. For predicting the final destination, the 3-gram model was most accurate at 71.56%, with accuracy decreasing as 'n' increased beyond 3. However, when predicting the endpoint mere minutes before arrival, the trend shifted. Larger n sizes proved more accurate, with the 6-gram model outperforming others at 43% accuracy for 2.5 minutes before arrival. The distinction in performance is attributed to the unique application of n-gram in the two scenarios. The latter allowed the model to harness recent trajectory data, making imminent destination predictions more precise.

When predicting the last destination in a trajectory using an LSTM model, 70% of the dataset was used for training, achieving an accuracy of 80.95%, while validation and test accuracies were 80.86% and 79.8%, respectively. For predictions minutes before arrival, a sliding window technique transformed each trajectory into fixed-length sub-sequences. Based on the explanation insights, the optimal window size was determined to be 6. When testing the model to predict the endpoint 2.5 to 6.25 minutes in advance, accuracies ranged from 18.16% to 47.92%, with higher accuracy as the endpoint neared. This model's strength lies in its ability to make sequential predictions, becoming more precise as the trajectory's end approaches.

The LSTM model, renowned for its capacity to process sequences, exhibited a distinct inclination towards recent data points in a trajectory. This observation was the impetus behind our novel training methodology. Employing the sliding window technique, we segmented primary trajectories into more concise sub-sequences. This methodological pivot not only aligned with the LSTM's inherent behavior but also amplified its predictive

capabilities. The culmination of this effort was evident in the LSTM's exceptional accuracy rates: 92.25% for predicting the next and final destination and 47.92% for predicting a stop just 2.5 minutes before its arrival.

While these predictive accomplishments were noteworthy, the thesis also underscored the significance of model explainability. In our pursuit to demystify the LSTM's decision-making process, tools like the LIME Text Explainer and perturbation tests were invaluable. They illuminated the model's reliance on the concluding trajectory points, reinforcing their pivotal role in the prediction process.

In essence, our journey in this thesis has been twofold. Firstly, it was about harnessing the power of machine learning models, like the LSTM, to predict trajectories with commendable accuracy. Secondly, it was about venturing beneath the surface, using tools of explainability, to understand the 'why' behind these predictions. This holistic approach not only strengthened the validity of our predictions but also laid a solid foundation for future research in trajectory prediction, emphasizing the balance between accuracy and comprehensibility.

Bibliography

[1] Yichuan Wang, Mengran Xiong, and Hossein Olya. Toward an understanding of responsible artificial intelligence practices. In *Proceedings of the 53rd hawaii international conference on system sciences*, pages 4962–4971. Hawaii International Conference on System Sciences (HICSS), 2020.

[2] Tobias Clement, Nils Kemmerzell, Mohamed Abdelaal, and Michael Amberg. Xair: A systematic metareview of explainable ai (xai) aligned to the software development process. *Machine Learning and Knowledge Extraction*, 5(1):78–108, 2023.

[3] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[4] Damien Garreau and Ulrike Luxburg. Explaining the explainer: A first theoretical analysis of lime. In *International conference on artificial intelligence and statistics*, pages 1287–1296. PMLR, 2020.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arxiv 2020. *arXiv preprint arXiv:2010.11929*, 2010.

[9] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4316–4336, 2020.

[10] Spyros Fountas, Borja Espejo-Garcia, Aikaterini Kasimati, Nikolaos Mylonas, and Nicoleta Darra. The future of digital agriculture: technologies and opportunities. *IT professional*, 22(1):24–28, 2020.

[11] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[12] Poornima Ramaswamy, James Jeude, and Jerry A Smith. Making ai responsible and effective, 2018.

[13] PwC. Responsible AI: Maturing from theory to practice, 2023. [Online; accessed on 27-July-2023].

[14] European Parliament and of the Council; Council of the European Union. Regulation (EU) 2016/679. European Union Law, 2016.

[15] Michael Van Lent, William Fisher, and Michael Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the national conference on artificial intelligence*, pages 900–907. Citeseer, 2004.

[16] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[18] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[19] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[20] Federico Cabitza, Andrea Campagner, and Davide Ciucci. New frontiers in explainable ai: understanding the gi to interpret the go. In *Machine Learning and Knowledge Extraction: Third IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2019, Canterbury, UK, August 26–29, 2019, Proceedings 3*, pages 27–47. Springer, 2019.

[21] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[22] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[24] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[25] Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298:103502, 2021.

[26] Sheng Wang, Zhifeng Bao, J Shane Culpepper, and Gao Cong. A survey on trajectory data management, analytics, and learning. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

[27] Maike Buchin, Anne Driemel, Marc J van Kreveld, and Vera Sacristán. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *Journal of Spatial Information Science*, 3:33–63, 2011.

[28] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. Calibrating trajectory data for similarity-based analysis. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*, pages 833–844, 2013.

[29] Kun Ouyang, Reza Shokri, David S Rosenblum, and Wenzhuo Yang. A non-parametric generative model for human trajectories. In *IJCAI*, volume 18, pages 3812–3817, 2018.

[30] Jingyuan Wang, Ning Wu, Xinxi Lu, Wayne Xin Zhao, and Kai Feng. Deep trajectory recovery with fine-grained calibration using kalman filter. *IEEE Transactions on Knowledge and Data Engineering*, 33(3):921–934, 2019.

[31] Renchu Song, Weiwei Sun, Baihua Zheng, and Yu Zheng. Press: A novel framework of trajectory compression in road networks. *arXiv preprint arXiv:1402.1546*, 2014.

[32] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343, 2009.

[33] Wannes Meert and Mathias Verbeke. Hmm with non-emitting states for map matching. In *European Conference on Data Analysis (ECDA), Date: 2018/07/04-2018/07/06, Location: Paderborn, Germany*, 2018.

[34] B. C. I. GmbH. bmwcarit/barefoot: Java library, 2015. Available at `https://github.com/bmwcarit/barefoot`.

[35] Viterbi algorithm - wikipedia. `https://en.wikipedia.org/wiki/Viterbi_algorithm`. Accessed: yyyy-mm-dd.

[36] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. Modeling trajectories with recurrent neural networks. IJCAI, 2017.

[37] Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Crailtap. Taxi trajectory dataset. `https://www.kaggle.com/datasets/crailtap/taxi-trajectory`, 2017.

[40] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[41] OpenStreetMap contributors. Openstreetmap, 2023. Accessed: 2023-07-27.

[42] Jinmeng Rao, Song Gao, Yuhao Kang, and Qunying Huang. Lstm-trajgan: A deep learning approach to trajectory privacy protection. *arXiv preprint arXiv:2006.10521*, 2020.

[43] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, 2005.

[44] Yaguang Tao, Alan Both, Rodrigo I Silveira, Kevin Buchin, Stef Sijben, Ross S Purves, Patrick Laube, Dongliang Peng, Kevin Toohey, and Matt Duckham. A comparative analysis of trajectory similarity measures. *GIScience & Remote Sensing*, 58(5):643–669, 2021.

[45] Raj Sangani. Interpreting an lstm through lime. `https://towardsdatascience.com/interpreting-an-lstm-through-lime-e294e6ed3a03`, 2021.